

Département Informatique
Institut Universitaire de Technologie
Université de Montpellier II
Montpellier, France

Pierre Feuille Ciseaux Lézard Spock

Rapport de projet tutoré



Auteurs :
Julien Dumartinet
Gaël Foppolo
Loïc Forthoffer
Pierre Marais

Tuteur :
Madalina Croitoru

www.pfcls.me

Année universitaire 2014 - 2015

Remerciements

Avant toute chose, nous souhaitons remercier notre tutrice, Madalina Croitoru, pour nous avoir permis de participer à ce projet ainsi que pour son implication constante tout au long du projet.

Dès le début, elle a su, grâce à sa rigueur scientifique, nous aider à nous poser les bonnes questions et à orienter nos recherches.

Très ouverte d'esprit, sa patience, ses conseils éclairés et sa disponibilité ont fait d'elle une valeur rassurante durant tout le projet.
Nous souhaitons aussi remercier ses étudiants de licence, qui, en jouant, nous ont permis de récolter une grande quantité de données et ainsi permis d'avoir des résultats plus pertinents.

Nous voulons également remercier Jérémy Cregut pour ses conseils, ses remarques et ses critiques qui nous ont été utiles tout au long de l'écriture de ce rapport.

Enfin nous tenons aussi à remercier Julien Rabatel pour nous avoir aidés dans nos choix de conceptions de notre projet, notamment grâce à sa thèse.

Table des matières

Table des figures

Glossaire

draw équivaut à une partie nulle.

framework ensemble de composants logiciels structurels qui servent à créer les fondations ainsi que les grandes lignes de l'architecture d'un logiciel.

granularité de l'information niveau de détails contenus dans une unité d'information. Plus il y a de détails, plus bas sera le niveau de la granularité. Inversement, moins il y a de détails, plus haut sera le niveau de la granularité.

intelligence artificielle programme affrontant le joueur en jouant de façon optimale.

motifs contextuels informations contextuelles associées aux données pour extraire des motifs fréquents représentatifs d'un contexte.

responsive une notion de conception Web qui vise à l'élaboration de sites offrant une expérience de lecture et de navigation optimales pour l'utilisateur quelle que soit l'appareil utilisé (ordinateur, tablette, smartphone...).

user friendly qui est facile à utiliser, ergonomique.

Chapitre 1

Introduction

Le jeu *Pierre Feuille Ciseaux* est un jeu non coopératif fondamental, opposant deux joueurs, où il faut affronter son adversaire à l'aide de trois « figures » que sont la pierre, la feuille et les ciseaux et en suivant les règles suivantes : la pierre casse les ciseaux, les ciseaux coupent la feuille et la feuille enveloppe la pierre. On remarque qu'aucune figure n'est absolument meilleure que les autres, chacune bat une figure et est battue par l'autre figure.

Le jeu dans cette forme basique souffre d'une limite assez flagrante : la probabilité d'obtenir un draw est $1/3$; il n'y a que trois combinaisons de figures qui aboutissent à un résultat gagnant/perdant. Pour pallier ce problème, nous avons choisi d'utiliser une variante : *Pierre Feuille Ciseaux Lézard Spock*. Créé par *Sam Kass* et popularisée par la série américaine *The Big Bang Theory*, il ajoute deux nouvelles figures au jeu, dans sa forme normale, ce qui a pour effet d'amener le nombre de combinaisons qui aboutissent à un résultat gagnant/perdant à dix, ainsi que la probabilité d'obtenir un draw à $1/5$. En plus des règles classiques, s'appliquent ici de nouvelles règles :

- la pierre écrase le lézard
- le lézard empoisonne Spock
- Spock casse les ciseaux
- les ciseaux décapitent le lézard
- le lézard mange la feuille
- la feuille désavoue Spock
- Spock vaporise la pierre

Le projet consiste à implémenter une interface web permettant à deux joueurs humains de jouer l'un contre l'autre ainsi que la possibilité de jouer contre une intelligence artificielle (IA). Les séquences de coups seront stockées à chaque partie pour permettre l'élaboration de statistiques de jeu.

La façon dont les humains prennent des décisions lors d'interactions stratégiques non coopératives est une question à laquelle il est difficile de répondre à l'heure actuelle. Pour un modèle aussi fondamental que ce jeu, on pourrait penser que ce serait uniquement une affaire de chance et de hasard et ainsi, que chaque joueur choisissait de manière aléatoire l'une de ces trois figures ou même si elles apparaissaient à intervalles réguliers au cours d'une partie, suivraient une logique d'équilibre statistique. Or, de nombreuses expériences suggèrent qu'il est quasi impossible pour un humain de générer une séquence aléatoire, chaque joueur est amené à générer son propre modèle de jeu, consciemment ou non [5]. Partant de ce constat, les joueurs essaient de comprendre les séquences de coups de leur adversaire pour s'y opposer et trouver un modèle émergent [1]. C'est dans ce cadre que s'inscrit ce projet. Il s'agit d'établir des statistiques sur l'utilisation des différentes figures. Grâce à ces statistiques, une IA sera mise en place pour

implémenter des stratégies de jeu et pour permettre à l'humain de jouer contre cette IA.

Ce projet d'un point de vue général, s'intéresse à la découverte de motifs contextuels dans les séquences de coups de façon à mettre en lumière des corrélations cachées dans les données ou des tendances de jeu générales [3]. En partant de l'idée générale qu'un motif fréquent représente un comportement attendu, nous nous posons les questions suivantes : *« Peut-on prévoir, à partir d'un motif fréquent et de données contextuelles, le prochain coup qui sera joué ? »* et *« Peut-on déduire, à partir d'un motif fréquent et de données contextuelles, une tendance de jeu ? »*

Tout d'abord, nous présenterons le cahier des charges du projet, contenant l'ensemble des besoins fonctionnels et des spécifications techniques, puis dans une deuxième partie, nous détaillerons dans le rapport technique les choix de conception que nous avons faits pour développer le jeu, les résultats obtenus accompagnés de leur critique. Vous trouverez également dans ce document un manuel d'utilisation à destination des utilisateurs de l'interface web, ainsi qu'un rapport d'activité rendant compte des méthodes de travail que nous avons utilisées pour conduire ce projet.

Chapitre 2

Cahier des charges

2.1 Introduction

Le projet *PFCLS* a pour but d'implémenter le jeu *Pierre Feuille Ciseaux Lézard Spock*. Les règles de jeu sont résumées dans la figure 2.1 :

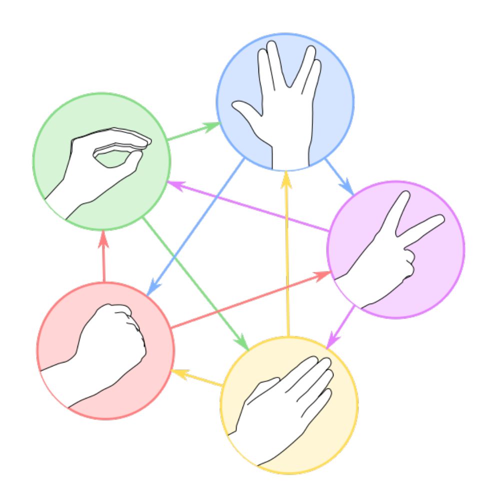


Figure 2.1 – Schéma des règles du jeu Pierre Feuille Ciseaux Lézard Spock

2.1.1 Objectifs

Cette partie présente les spécifications des exigences logicielles du projet open source *Pierre Feuille Ciseaux Lézard Spock*. Il peut être utilisé comme base pour une possible extension du projet actuel. Ce cahier des charges est rédigé sous le standard IEEE 830 [4].

2.1.2 Conventions du document

PFCLS a été développé pendant l'écriture de la présente partie, donc certaines exigences stipulées ici ne sont pas encore satisfaites. Il est très important de mettre à jour cette partie avec toutes les futures exigences et de clarifier chaque élément à des fins de cohérence, de sorte que cette partie puisse rester utile.

Certaines informations techniques ont été incluses, d'autres non, les lecteurs peuvent se référer au glossaire disponible en début de document pour trouver les définitions des termes spécifiques.

2.1.3 Public visé

Cette partie est destinée :

- aux développeurs qui peuvent examiner les capacités du projet et plus facilement comprendre où leurs efforts devraient être ciblés pour améliorer ou ajouter des fonctionnalités au projet (il établit les lignes directrices pour un développement futur)
- aux testeurs qui peuvent utiliser cette partie comme base pour leur stratégie de tests, certains bugs peuvent être plus faciles à déceler à l'aide du cahier des charges
- aux utilisateurs finaux de l'application qui souhaitent en savoir plus sur ce que ce projet peut faire

2.2 Description générale

Cette section donnera un aperçu de l'ensemble du système. Le système sera expliqué dans son contexte pour montrer comment le système interagit avec les utilisateurs et les fonctionnalités principales seront introduites. Enfin, les contraintes et les hypothèses pour le système seront présentées.

2.2.1 Perspective du produit

Le système va devoir communiquer avec l'utilisateur afin de récupérer les données, les traiter et enfin afficher les résultats. Il devra être capable de récupérer les coups joués et d'afficher les résultats. Le jeu *Pierre Feuille Ciseaux Lézard Spock* ne fonctionne pas de façon indépendante. Comme c'est un produit centré sur les données, il devra avoir un endroit pour les stocker. Pour ça une base de données devra être utilisée pour stocker ou récupérer les données de joueurs ou de jeu. Toutes les communications à la base de données se feront par Internet.

Le système de notre projet comporte 1 acteur et 2 systèmes qui coopèrent. Le joueur est notre seul acteur. Le système principal (gestion des utilisateurs, système de jeu et IA) et la base de données sont les 2 systèmes qui coopèrent. Chaque acteur accède au système par Internet et communique avec la base de données via le système principal, voir figure 2.2.
Dans la suite du cahier des charges, les acteurs seront désignés par le terme joueurs et le système de notre projet sera désigné par le terme application.

2.2.2 Fonctions du produit

L'application doit offrir deux fonctionnalités principales à un joueur : jouer contre un autre joueur et jouer contre le système (IA).
Le système principal doit interagir avec le joueur pour récupérer les coups qu'il joue, les transmettre à la base de données et afficher les résultats intermédiaires et finaux. Il doit aussi gérer les cas d'erreurs lors d'une tentative d'accès à une partie non autorisée de l'application et la synchronisation avec la base de données pour toujours être dans un état cohérent. La figure 2.3 détaille de façon exhaustive les fonctionnalités auxquelles le joueur a accès.

2.2.3 Caractéristiques des utilisateurs

Ce projet ne se réfère qu'à un seul type d'utilisateur : le joueur. Celui-ci interagit avec l'application via une interface web. Cette interface doit être *user friendly*, intuitive et accessible.

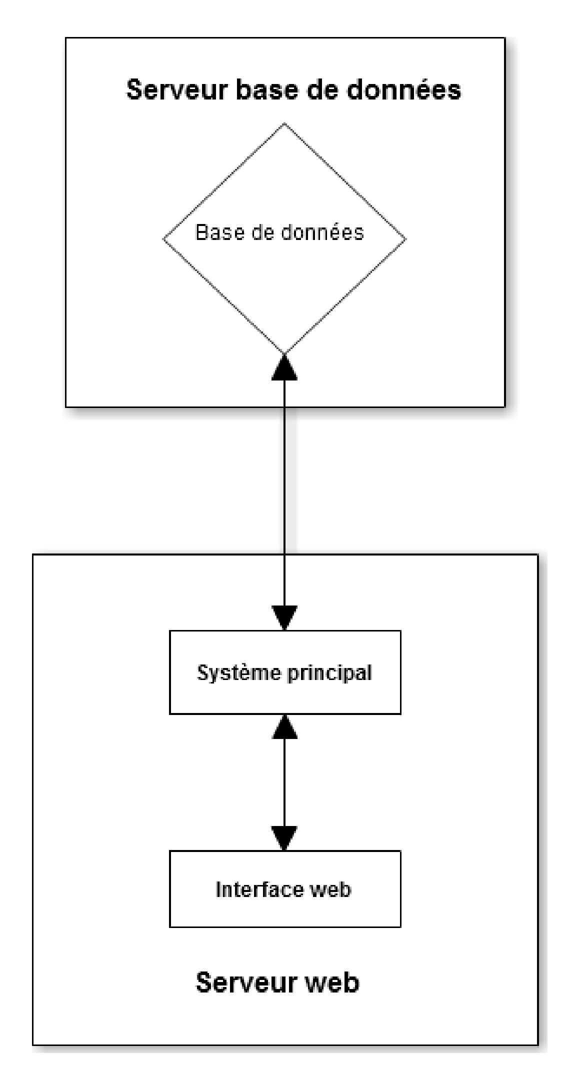


FIGURE 2.2 – Schéma fonctionnel (block diagram)

2.2.4 Environnement de fonctionnement

PFCLS est une application indépendante d'un point de vue du système, mais qui a besoin d'un navigateur et d'une connexion Internet pour fonctionner. L'application a été testée sous les navigateurs suivants :

- Firefox 33
- Chrome 38
- Opera 26
- Safari 8
- Internet Explorer 11

Ainsi que sous les navigateurs mobiles suivants :

- Safari (iOS 8)
- Chrome (iOS 8, Android 4.1)
- Internet Explorer Mobile (Windows Phone 8.10)

Aucune autre condition n'est requise pour faire fonctionner l'application.

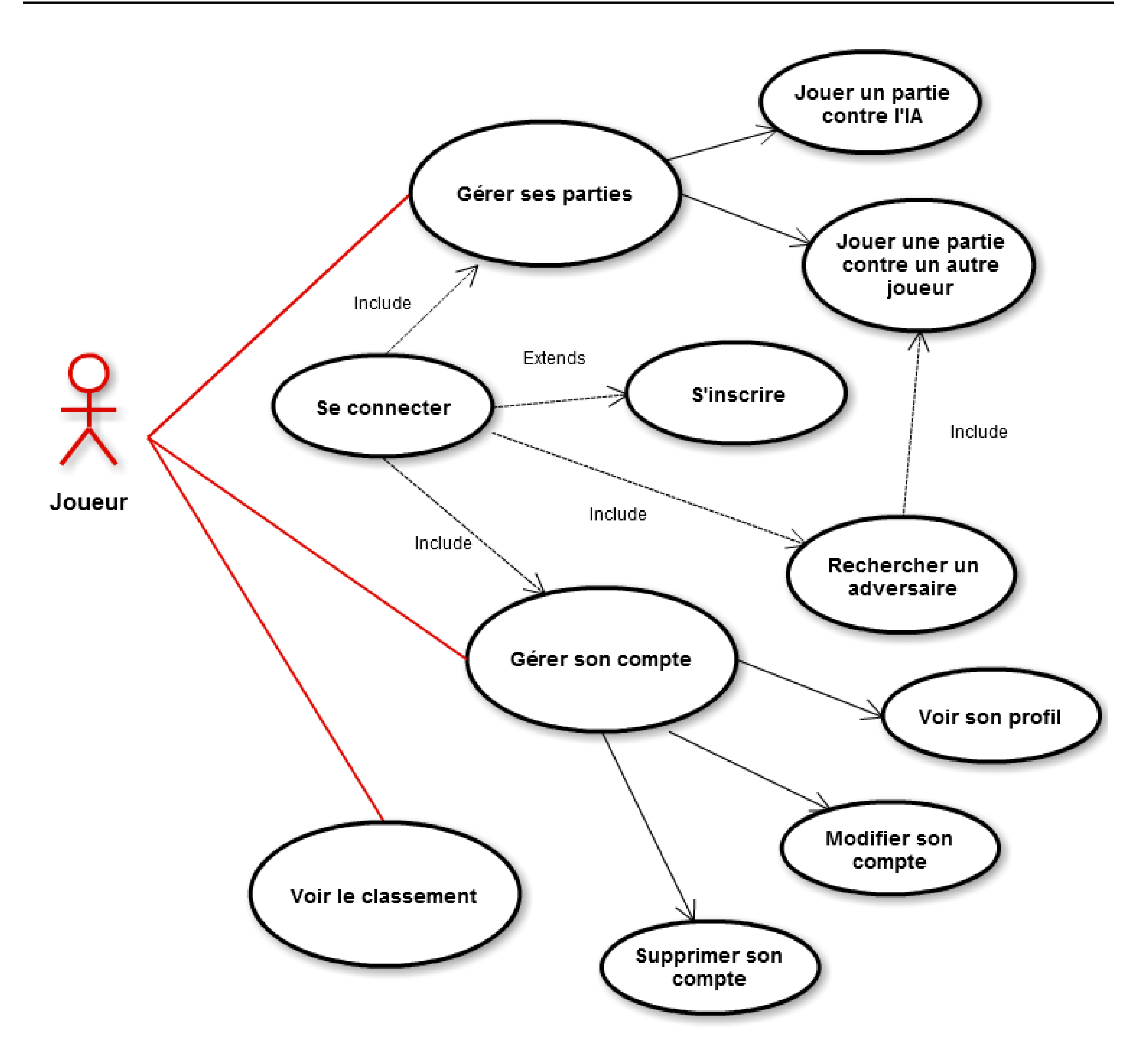


FIGURE 2.3 – Diagramme des cas d'utilisations

2.2.5 Contraintes de conception et d'implémentation

PFCLS est une application indépendante développée en PHP/SQL avec une interface utilisateur écrite en HTML/CSS/JS.
Le projet est sous licence *GNU/GPLv3* [2]. Tous ceux qui souhaitent utiliser ou travailler sur le projet doivent accepter pleinement les termes de ce type de licence.

L'interface web de l'application impose de fortes contraintes au projet. En effet, comme il existe plusieurs navigateurs, chacun avec ses propres caractéristiques, l'interface ne sera pas exactement la même pour tous les utilisateurs. Des différences d'affichage peuvent survenir.

La connexion Internet est aussi une exigence pour l'application. Comme celle-ci récupère et stocke les données dans une base de données via Internet, il est crucial d'avoir une connexion Internet pour que l'application fonctionne.

2.2.6 Hypothèses et dépendances

Une connexion Internet stable (minimum ADSL) est requise pour que l'application fonctionne correctement. Si la connexion est peu stable, trop faible, il se peut que le joueur rencontre des

difficultés à se connecter à l'interface web ou à jouer une partie complète sans être déconnecté.

Une autre dépendance est le navigateur web. Si celui-ci n'est pas à jour, il se peut que l'interface web ne soit pas totalement fonctionnelle, qu'elle offre des fonctionnalités réduites ou qu'elle soit complètement non opérationnelle.

2.3 Interfaces externes

2.3.1 Interface joueurs (*GUI*)

Lors de sa première utilisation, un joueur devra se connecter pour accéder aux fonctionnalités du système. Si le joueur n'est pas enregistré, il devra s'inscrire.

Si ce n'est pas sa première utilisation, c'est-à-dire s'il est connecté, la page par défaut sera la page de choix de jeu. Il doit aussi voir le menu qui liste toutes les actions disponibles, voir figure 2.4.

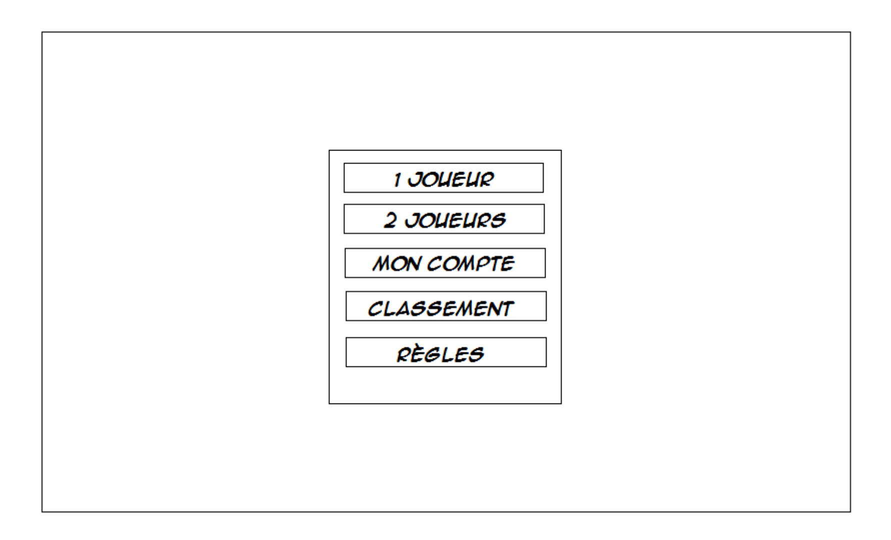


FIGURE 2.4 – Croquis du menu principal

Chaque joueur doit avoir un profil où il peut modifier ses informations (pseudo, mot de passe, etc.) et supprimer son propre compte. Il peut aussi consulter le classement.

Dans la figure 2.4, on peut voir le choix possible entre une partie contre l'IA (un joueur) ou contre un autre joueur (deux joueurs). Quand le joueur choisit une partie contre l'IA, il lui ai demandé de saisir un nombre de manches et la partie démarre immédiatement. En revanche, quand le joueur choisit une partie contre un autre joueur, le joueur voit la liste des joueurs en attente d'un adversaire et doit aussi avoir la possibilité de saisir un nombre de manches.

Dans la vue de choix de la figure à jouer lors d'une partie, chaque joueur devra choisir la figure qu'il souhaite en ayant simplement à cliquer dessus. Chaque figure est représentée par une image avec son nom, voir figure 2.5.

La vue de résultat du coup montrera les deux figures jouées ainsi que la vainqueur de la manche, voir figure 2.6.

La vue de résultat de la partie affiche le nom du gagnant, le score ainsi qu'un petit message, voir figure 2.7.

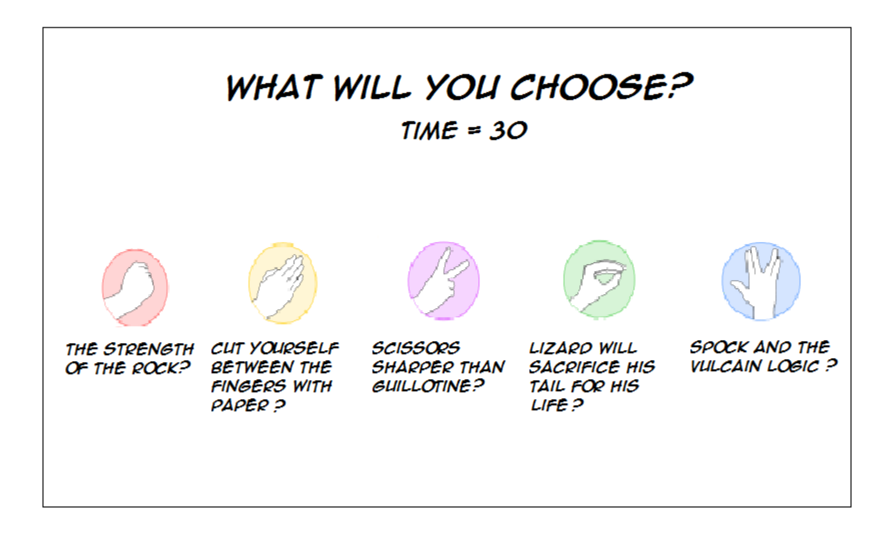


FIGURE 2.5 – Croquis du choix de la figure à jouer

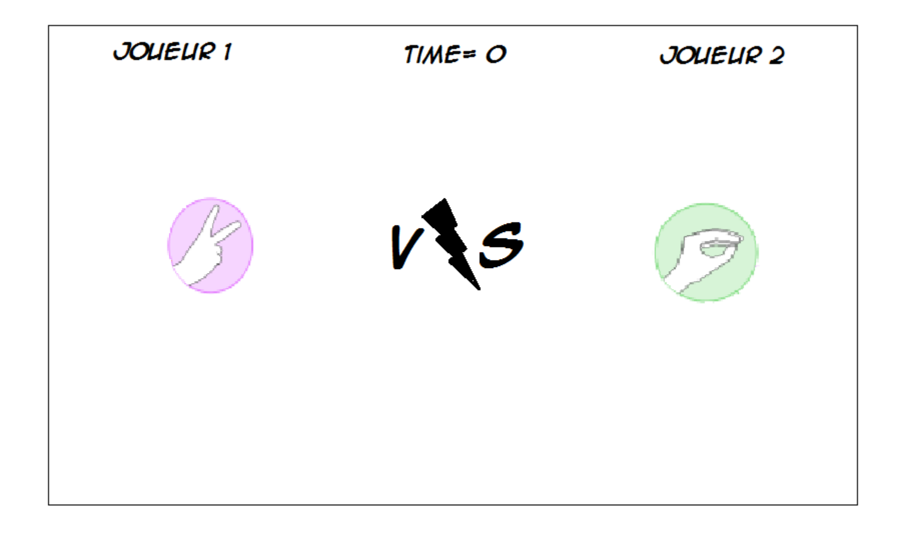


FIGURE 2.6 – Croquis du résultat de la manche



FIGURE 2.7 – Croquis du résultat de la partie

2.3.2 Interfaces matérielles

L'utilisateur peut accéder à l'application à partir d'un ordinateur, d'un smartphone ou d'une tablette.

2.3.3 Interfaces logicielles

Le système principal communique avec la base de données afin d'obtenir des informations sur les joueurs, le classement, le jeu ou les parties pour les représenter sous forme graphique au joueur, voir figure 2.2. La communication entre la base de données et le système principal se compose d'opérations de lecture et d'écriture des données. Il n'y aucune communication directe entre le joueur et la base de données.

2.3.4 Protocoles et interfaces de communications

La communication entre les deux parties de l'application est importante, car chacune dépend de l'autre. Cependant, la manière dont la communication est réalisée n'est pas importante, car c'est le système principal lui-même qui gère la communication.
Comme notre application repose sur la base de données, il y a de très nombreux échanges avec celles-ci. Pour que l'application puisse supporter un nombre conséquent de requêtes simultanées et qu'elle assure sa cohérence à chaque instant, elle doit être suffisamment puissante. C'est donc la puissance du serveur où est hébergé la base de données qui déterminera le nombre de transactions maximales par seconde qu'elle peut traiter ainsi que le nombre maximal de joueurs simultanés possibles sur l'interface web.

2.3.5 Fonctionnalités du système

2.3.6 Inscription utilisateur

L'utilisateur doit être en mesure de s'inscrire grâce à l'interface web. Il doit fournir un pseudo, un mot de passe, une adresse e-mail, son sexe et son âge.

- **ID :** I1
- **Nom :** « Inscription »
- **Type :** Formulaire
  - (I11) « Pseudo » : zone de texte
  - (I12) « Sexe » : bouton radio
  - (I13) « Âge » : zone de texte (entier)
  - (I14) « Mot de passe » : zone de texte
  - (I15) « Confirmation de mot de passe » : zone de texte
  - (I16) « E-mail » : zone de texte (e-mail)
- **Prérequis :** Ne pas être connecté sur le site
- **Postrequis :** tous les champs sont bien remplis, les deux mots de passe sont identiques et le pseudo/e-mail n'est pas déjà utilisé

2.3.7 Connexion utilisateur

L'utilisateur doit être en mesure de se connecter grâce à l'interface web. Il doit fournir un pseudo et un mot de passe.

- **ID :** C1
- **Nom :** « Connexion »

- **Type :** Formulaire
  - (C11) « Pseudo » : zone de texte
  - (C12) « Mot de passe » : zone de texte
- **Prérequis :** Ne pas être connecté sur le site
- **Postrequis :** tous les champs sont bien remplis et correspondent aux données de la base de données

2.3.8 Profil utilisateur

La page du profil utilisateur affiche trois onglets, chacun permettant de voir une partie précise du profil :

- Onglet *Profil* (O1) affiche les informations personnelles du joueur
- Onglet *Statistiques de jeu* (O2) affiche les statistiques de jeu du joueur
- Onglet *Historique des parties* (O3) affiche la liste des 10 dernières parties jouées par le joueur

Il est à noter que lorsque le joueur arrive sur la page de profil, c'est l'onglet *Profil* (O1) qui s'affiche par défaut.

- **ID :** 01
- **Nom :** « Profil »
- **Type :** Bouton
- **Prérequis :** section 2.3.7
- **Action :** affiche les informations personnelles du joueur, pseudo, sexe, âge, classement, ratio ainsi qu'un lien pour modifier son profil et le supprimer
- **Postrequis :** *N/A*

- **ID :** 02
- **Nom :** « Statistiques de jeu »
- **Type :** Bouton
- **Prérequis :** section 2.3.7
- **Action :** affiche les statistiques de jeu du joueur, comme les coups qu'ils jouent le plus en premier, etc.
- **Postrequis :** *N/A*

- **ID :** 03
- **Nom :** « Historique des parties »
- **Type :** Bouton
- **Prérequis :** section 2.3.7
- **Action :** affiche sous forme d'une liste les dernières parties jouées par le joueur, avec le gagnant et le score
- **Postrequis :** *N/A*

Modifier son profil utilisateur

L'utilisateur doit être en mesure de modifier ses informations personnelles, comme son pseudo, son mot de passe ou son âge. Il doit fournir un pseudo et un e-mail qui ne sont pas déjà utilisé par un autre joueur.

- **ID :** P1

- **Nom :** « Modifier profil »
- **Type :** Formulaire
  — (P11) « Pseudo » : zone de texte
  — (P13) « Âge » : zone de texte (entier)
  — (P14) « Mot de passe » : zone de texte
  — (P16) « E-mail » : zone de texte (e-mail)
- **Prérequis :** section 2.3.7
- **Action :** modifie le profil du joueur avec les informations saisies
- **Postrequis :** tous les champs sont bien remplis et le pseudo/e-mail n'est pas déjà utilisé

Supprimer son profil utilisateur

L'utilisateur doit être en mesure de supprimer son profil, c'est-à-dire que toutes les informations de son profil seront supprimées. Il n'aura plus la possibilité de se connecter.

- **ID :** P2
- **Nom :** « Supprimer son profil »
- **Type :** Bouton
- **Prérequis :** section 2.3.7
- **Action :** supprime le profil du joueur de façon définitive
- **Postrequis :** *N/A*

2.3.9 Classement

L'utilisateur doit être en mesure de voir le classement général du site, sous la forme d'un tableau, classé selon le ratio, de façon décroissante. Chaque élément de la liste représente un joueur, il doit inclure le classement, le pseudo et le ratio. Aucune limite d'affichage n'est requise pour le moment.

- **ID :** C1
- **Nom :** « Classement »
- **Prérequis :** *N/A*
- **Action :** affiche sous forme de liste le classement des joueurs
- **Postrequis :** *N/A*

2.3.10 Rechercher un adversaire

L'utilisateur doit être en mesure de rechercher un adversaire pour jouer une partie. Dans ce cas, il peut directement choisir son adversaire dans la liste des adversaires en attente ou alors choisir un nombre de manches à jouer et se mettre en attente d'un adversaire. Il ne peut y avoir deux joueurs en attente d'un adversaire avec le même nombre de manches ; si c'est le cas, les deux joueurs sont connectés pour jouer la partie.

- **ID :** R1
- **Nom :** « Rechercher adversaire »
- **Type :** Bouton
- **Prérequis :** section 2.3.7 et ne pas être déjà en cours de partie
- **Action :** lance la partie si l'adversaire est trouvé directement ou met en attente d'un adversaire
- **Postrequis :** *N/A*

2.3.11 Jouer une partie contre un autre joueur

L'utilisateur doit être en mesure de jouer une partie contre un autre joueur. Quand il a trouvé son adversaire, la partie est lancée et le joueur est invité à chaque tour à choisir la figure qu'il souhaite jouer.

- **ID :** J1
- **Nom :** « Jouer contre un joueur »
- **Prérequis :** section 2.3.7 et section 2.3.10
- **Action :** choisir une figure et affiche le résultat à chaque tour
- **Postrequis :** la figure choisie est valide

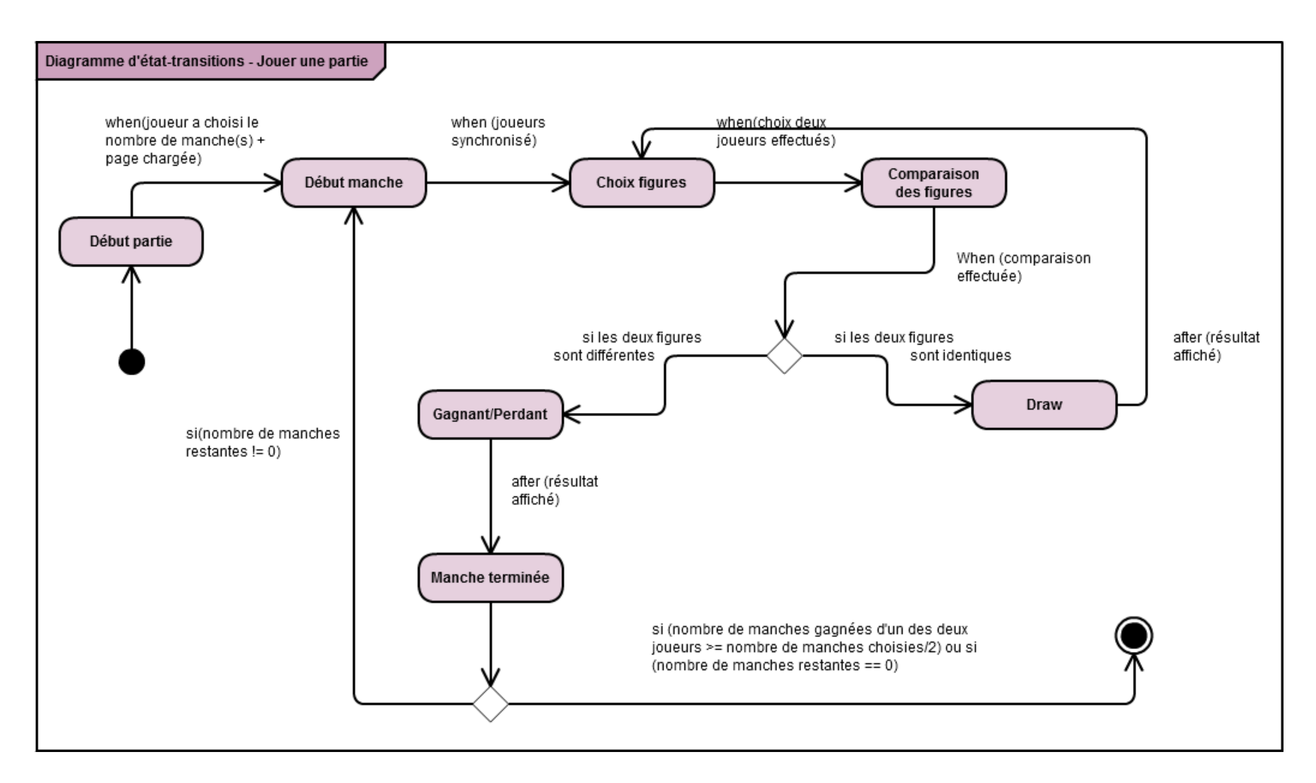


FIGURE 2.8 – Diagramme d'états-transitions *Jouer une partie contre un autre joueur*

Les diagrammes d'états-transitions (cf figure 2.8) et d'activité (cf figure 2.9) *Jouer une partie contre un autre joueur*, détaillent de façon plus explicite la fonctionnalité 2.3.11. Le joueur choisit le nombre de manches puis attend le chargement de la page. Quand le joueur est synchronisé, il peut choisir la figure qu'il va jouer. Le système récupère alors son choix et le compare avec celui de son adversaire, puis il affiche le résultat : gagnant, perdant ou draw. Si c'est un draw, on invite les deux joueurs à choisir à nouveau une figure. Sinon, il signale la manche comme terminée à la base de données. S'il reste encore des manches à jouer ou qu'il n'y a pas de gagnant, on invite le joueur à choisir sa figure, etc. sinon on signale la partie terminée et on enregistre les statistiques dans la base de données.

2.3.12 Jouer une partie contre l'IA

L'utilisateur doit être en mesure de jouer une partie contre l'IA. Quand la partie est lancée, le joueur est invité à chaque tour à choisir la figure qu'il souhaite jouer.

- **ID :** J2

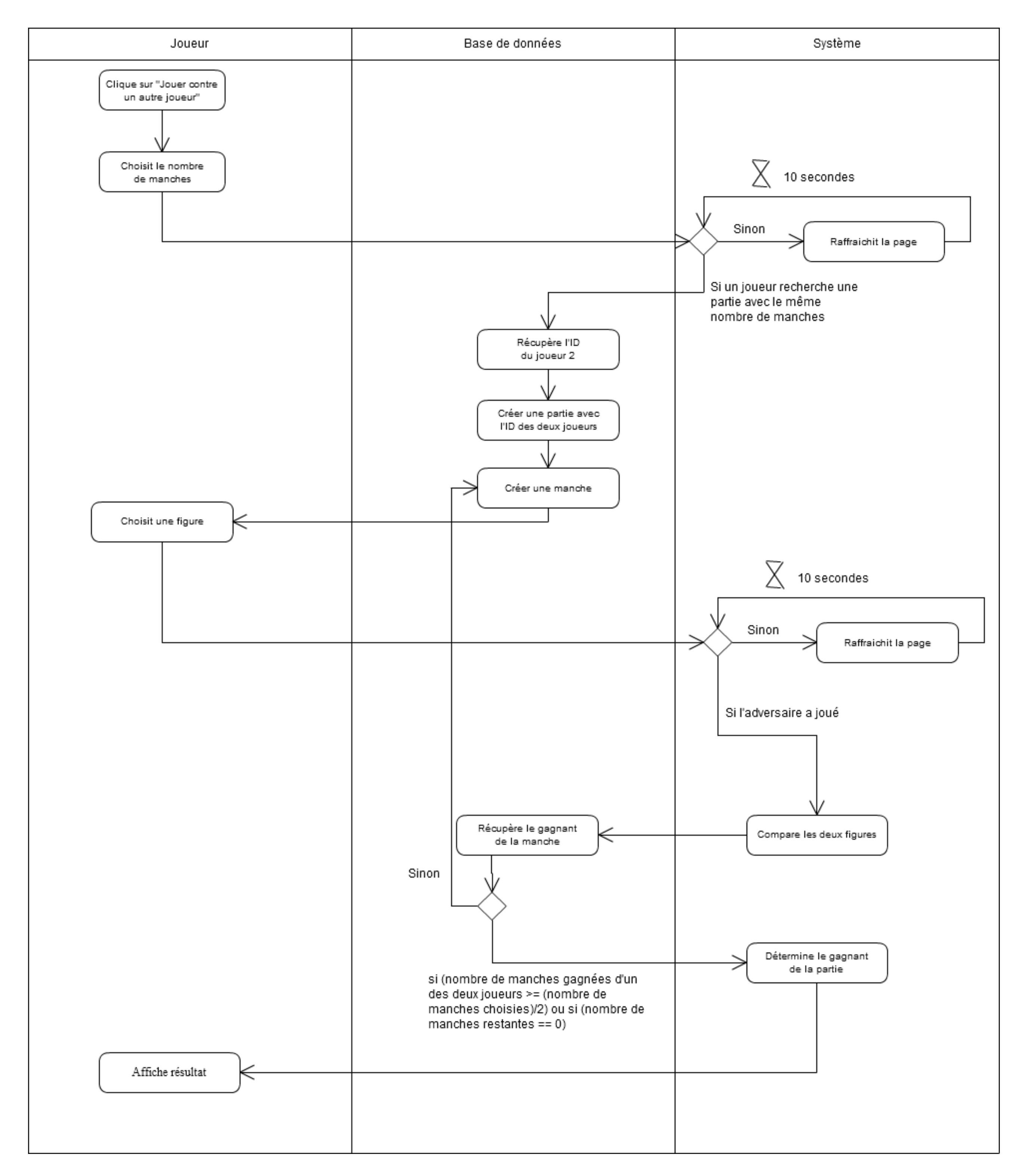


FIGURE 2.9 – Diagramme d'activité *Jouer une partie contre un autre joueur*

- **Nom :** « Jouer contre l'IA »
- **Prérequis :** section 2.3.7 et pas déjà en cours de partie contre l'IA
- **Action :** choisir une figure et affiche le résultat à chaque tour
- **Postrequis :** la figure choisie est valide

2.4 Exigences non fonctionnelles

2.4.1 Performance

PFCLS est une application qui n'a besoin que de quelques ressources pour fonctionner. Il a été conçu pour ne pas retarder le fonctionnement des autres processus clés du système. Le temps de réponse du programme dépend uniquement du serveur web et du serveur de base de données, l'application est considérée temps réel.

Néanmoins, une forte affluence de requêtes sur une des deux ressources fera automatiquement ralentir l'autre.

2.4.2 Exigences de sureté

L'application doit s'assurer que toutes les données saisies par les utilisateurs soient valides. De plus l'application doit, dans le cas d'une insertion de données invalides ou d'un accès non autorisé, fournir au joueur des messages d'erreur ou d'aide appropriés.

2.4.3 Exigences de sécurité

PFCLS n'introduit aucun niveau de sécurité, en effet les données gérées ne sont pas critiques. Les utilisateurs de l'application sont tous égaux donc il n'y a pas besoin d'avoir un niveau de permissions pour la gestion d'utilisateurs ou des données.

2.4.4 Attributs de la qualité logiciel

Cette application fournit une interface graphique agréable et *user-friendly* avec des fonctions relativement simples.Tout utilisateur doit être en mesure d'utiliser *PFCLS* sans aucune connaissance ou expérience spécifique. Les utilisateurs doivent seulement fournir des informations simples et en quelques clics, accèdent à toutes les fonctionnalités de l'application.

CHAPITRE 3

Rapport technique

3.1 Choix technologiques

Pour ce qui est de l'aspect technique du projet, certains choix technologiques ont été faits, mais ont finalement changé avant le début du développement. Ces choix étaient dans un premier temps :

- l'environnement bas niveau *Node.js* semblait être un bon choix pour le développement du jeu, car cet environnement facilite la synchronisation (le déroulement de la partie) entre les deux joueurs.
- l'utilisation du SGBD *MongoDB* avait été décidée, car c'est un système *NoSQL*, permettant de meilleures performances lors de la manipulation d'une grande quantité de données. Or pour la réalisation de l'IA du projet, un grand nombre de données doit être traité rapidement

Ces choix avaient également été motivés par le fait que ces deux technologies sont des technologies émergentes et qui tendent à se développer dans le milieu professionnel. Cependant, seuls les serveurs d'hébergement haut de gamme (dédiés) permettent l'utilisation de *Node.js* et *MongoDB*, c'est pour cette raison que des changements de stratégie technologique ont été fait. Ainsi, après une deuxième réunion avec notre tutrice, nous avons finalement décidé d'utiliser des technologies plus répandues :

- le Modèle Conceptuel de Données (MCD) pour la conception de la base de données
- le langage SQL pour construire cette base de données. Il servira également de langage de description des données.
- le langage PHP est utilisé pour le développement de l'application.
- les langages HTML5 & CSS3 pour la réalisation de l'interface web qui servira d'interface utilisateur.
- le langage JavaScript pour certaines animations de l'interface web.

Ces technologies ont été finalement choisies, car ce sont des technologies que nous avons étudiées au cours de notre formation, ainsi, nous serions opérationnels immédiatement et nous n'aurions pas besoin d'un temps d'apprentissage avant de pouvoir entamer le développement du projet, ce qui aurait été le cas si nous avions utilisé *Node.js* et *MongoDB*. PHP étant un langage très complet, il nous a permis de coder l'intégralité des fonctionnalités du jeu et du site. SQL répond parfaitement à nos besoins même si nous avons dû opérer quelques changements par rapport au choix des données à stocker de manière permanente ou temporaire ainsi qu'au développement des différents algorithmes.

Pour le développement de l'interface web, le framework Bootstrap a été choisi afin d'obtenir une interface *responsive*, rapide et ergonomique très facilement. Pour ce projet il semblait

évident pour toute notre équipe qu'il faudrait créer une interface web adaptée pour mobiles (smartphones, tablettes) c'est pour cette raison que ce framework a été choisi, il est simple d'utilisation, d'adaptation, de modification et permet d'obtenir un site *fully responsive*.

Concernant l'aspect pratique du projet, les principaux langages utilisés étant le PHP et le SQL, il a fallu trouver des outils qui répondent à nos besoins.
Nous nous sommes ainsi tournés pour le serveur de production vers l'utilisation d'un serveur Apache couplé à un serveur possédant MySQL comme « SGBD » et équipé de l'interface phpMyAdmin.

Quant au développement en local, nous avons opté pour l'utilisation du kit Wamp, logiciel permettant d'installer un serveur avec les mêmes outils que sur le serveur de production et nous permettant d'effectuer des tests dans des conditions relativement similaires aux conditions réelles.

Pour le développement en lui-même, nous avons utilisé de multiples éditeurs de texte comme *Atom*, *Sublime Text* ou *Notepadd++* qui ont été amplement suffisants pour notre projet.

3.2 Conception

3.2.1 Schéma relationnel des données

Pour démarrer la conception de projet, il a tout d'abord fallu extraire de l'analyse des besoins des informations sur le projet qu'il fallait mener. Le traitement de ces informations, de longs débats entre les membres du groupe et avec le client ont permis de définir le diagramme d'entité-association présenté dans la figure 3.1.
Une fois, le diagramme d'entité-association terminé, il a été possible de concevoir le modèle physique de données (MPD) qui consiste à implémenter le modèle conceptuel dans le SGBD, ici MySQL.

3.2.2 Architecture

Le projet est intégralement structuré selon le modèle MVC (Modèle Vue Contrôleur). Cette architecture a été adoptée, car c'est l'une des plus utilisées, nous l'avons étudié en cours et elle permet une organisation claire du projet.

- le **modèle** : cette partie gère les données du site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc les requêtes SQL.
- la **vue** : cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML ou du JavaScript, mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple la liste des joueurs en attente.
- le **contrôleur** : cette partie gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. Par exemple, c'est lui qui détermine si le joueur a le droit de voir la page ou non.

La figure 3.2 montre le fonctionnement d'un modèle MVC orienté web et l'annexe A montre l'architecture complète de notre projet.

Figure 3.1 – Diagramme entité-relation

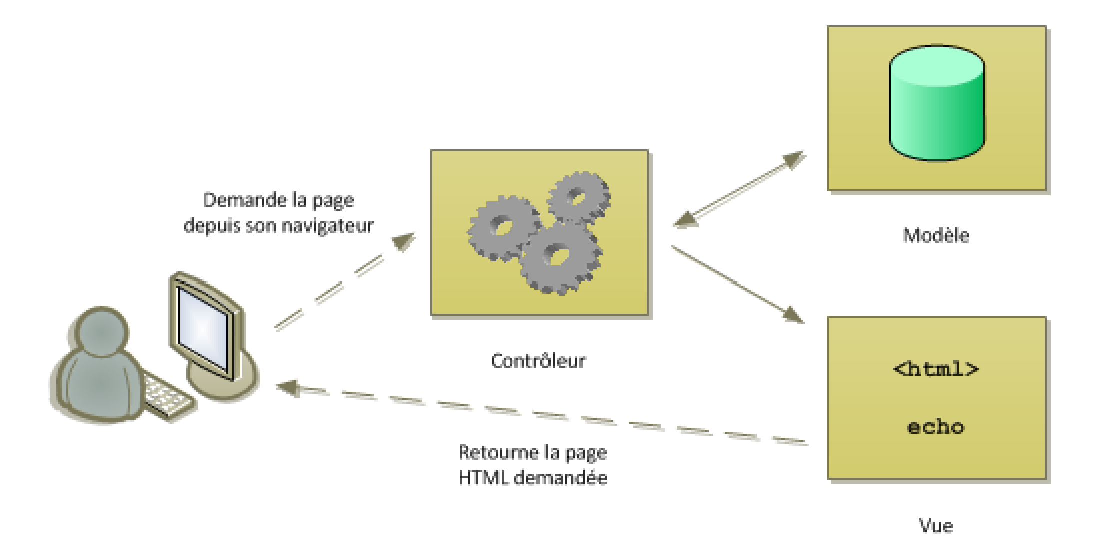


Figure 3.2 – Modèle MCV

3.3 Résultats du développement

Le site web a été développé en HTML, PHP et JavaScript. Il sera développé dans cette section le résultat du développement qui a pu être effectué. Les principales fonctionnalités dé-

veloppées décrites ci-dessous sont les fonctionnalités les plus importantes de notre projet.

Toutes les pages de l'interface web ont une mise en page et un design commun. Il y a un fichier *vue.php* qui gère le design du site (header et footer) et qui charge la vue correspondant à la page choisie par le contrôleur en cours. Ce contrôleur s'occupe de charger les données propres à chaque page consultée.

Nous détaillions uniquement deux fonctionnalités, les plus importantes de notre point de vue.

3.3.1 Recherche d'un adversaire

La recherche d'un adversaire pour jouer une partie est une fonction primordiale dans notre projet, elle permet de mettre en relation deux joueurs pour qu'ils puissent disputer une partie. L'algorithme est détaillé dans l'annexe B. La page de recherche d'un adversaire est appelée par le contrôleur *Jeu*.
Celui-ci gère les différents cas liés à l'état actuel du joueur (en partie, en recherche d'adversaire) par le biais de variables $_*SESSION* et $_*POST*.

3.3.1.1 Les cas d'erreurs

Il existe quatre cas d'erreurs qui sont traités par l'algorithme :

- le joueur est déjà présent dans une partie en cours.
- le joueur a déjà lancé une recherche d'adversaire.
- le nombre de manches n'a pas été indiqué par le joueur
- le joueur n'est pas connecté

Chacun de ces cas est traité grâce aux variables $_*SESSION* et $_*POST* qui nous permettent de garder en mémoire les informations relatives au joueur.

3.3.1.2 Le cas nominal

Le joueur est connecté et lance une recherche de joueur via la fonction **rechercheAdversaire(idJoueur, nbManches)** qui retourne l'*idJoueur* de l'adversaire ou NULL si elle n'en trouve aucun. Si un joueur est trouvé, on créée une partie avec les caractéristiques correspondantes et on dirige le joueur vers la vue pour débuter la parie. Si aucun n'est trouvé, on reste sur le page de recherche d'un adversaire.

3.3.1.3 Fonctionnement de l'algorithme

On commence par vérifier les quatre cas d'erreurs par ordre de priorité, par exemple, si le joueur n'est pas connecté, il ne peut pas avoir déjà lancé une recherche d'adversaire.
Après ces tests, on cherche les joueurs qui ont un nombre de manches similaire à celui passé en paramètre. Si on n'en trouve pas, on ajoute le joueur à la file d'attente. Toutes les 5 secondes, on vérifie que le joueur est toujours en attente. S'il n'y est plus, c'est qu'un adversaire a été trouvé, on l'envoie donc sur la page de choix des figures.
Si on en trouve, on supprime le joueur trouvé de la file d'attente et on crée une partie. On envoie ensuite le joueur sur la vue de sélection des figures.

3.3.1.4 Description des fonctions utilisées

- **rechercheAdversaire(*idJoueur*,*nbManches*)** : permet de rechercher si un autre joueur que *idJoueur* est présent dans la liste d'attente avec le même nombre de manches que *nbManches*.

- **ajouterAttente(*idJoueur*,*nbManches*)** : sert à ajouter un joueur à la file d'attente, on récupère le paramètre *nbManches* et *idJoueur* pour l'afficher dans la liste des joueurs en attente.
- **deletteAttente(*idJoueur*,*nbManches*)** : supprime le joueur de la file d'attente.
- **ajouterPartie(*idJoueur*, *idJoueurAdversaire*,*nbManches*)** : crée une partie entre les deux joueurs et avec le nombre de manches pendant lesquelles ils vont s'affronter.
- **ajouterManche(*idPartie*)** : créer une manche appartenant à la partie *idPartie*.
- **ajouterCoup(*idManche*, *idJoueur*,*idJoueurAdversaire*)** : crée un coup appartenant à la manche *idManche*, entre les deux joueurs.

3.3.2 Intelligence artificielle

L'intelligence artificielle a une très grande importance dans notre projet, car elle permet deux choses : d'une part, pour le joueur, la possibilité de jouer même s'il n'y a aucune autre personne à qui se mesurer et d'autre part, elle nous permet de vérifier le résultat de notre travail. En effet, plus l'IA gagne, plus cela signifie que notre travail porte ses fruits.
L'importance de l'intelligence artificielle réside aussi dans le fait qu'elle utilise les motifs contextuels dans sa recherche de la meilleure figure à jouer après la séquence que vient de jouer le joueur.
La fonction de l'intelligence artificielle est appelée par le contrôleur *JeuIA*.
Celui-ci gère les différents cas liés à l'état de la partie en cours (évaluation d'un coup, jouer, etc.) par le biais de variables $_*SESSION* et $_*POST*.
Nous traiterons ici seulement l'algorithme de l'intelligence artificielle (voir l'annexe C).

3.3.2.1 Le cas d'erreur

Seul un cas d'erreur existe, il s'agit du cas où il n'y a aucune donnée de jeu exploitable. L'IA choisit donc aléatoirement la figure à jouer.

3.3.2.2 Le cas nominal

Il s'agit du cas où il y a des données de jeu exploitables. On récupère ainsi toutes ces données de jeu sous la forme d'une liste des séquences de coups. De cette liste et avec la séquence de coups déjà joué lors de la partie en cours, on calcule la figure que le joueur à le plus de chance de jouer. Nous déduisons ensuite de ce résultat la figure à jouer, sortons de la boucle et retournons la figure à jouer.

3.3.2.3 Fonctionnement de l'algorithme

L'algorithme de l'intelligence artificielle est composé de plusieurs parties :
- recherche dans les données de jeu du joueur
- recherche dans les données de jeu des joueurs ayant le même sexe et le même âge (avec une marge de plus ou moins deux ans)
- recherche dans les données de jeu des joueurs ayant le même âge (avec une marge de plus ou moins deux ans), mais de sexe opposé
- recherche dans les données de jeu des joueurs ayant le même sexe et le même âge (avec une marge de plus ou moins cinq ans)
- recherche dans les données de jeu des joueurs ayant le même âge (avec une marge de plus ou moins cinq ans), mais de sexe opposé
- cas d'erreur (voir section 3.3.2.1)

Permier niveau de pertinence (joueur) C'est le niveau de pertinence le plus élevé, car la recherche est effectuée parmi les données de jeu du joueur. C'est ce qui permet d'approcher le plus la tendance de jeu qui lui est propre. Pour que l'intelligence artificielle soit la plus exacte possible, il serait souhaitable d'avoir une très grande quantité de données de jeu et un comportement de jeu « normal » et non un comportement de jeu « erratique ».

Deuxième niveau de pertinence (âge -/+ 2 ans) On passe au deuxième niveau de pertinence, si aucune donnée n'est exploitable dans les données de jeu du joueur. Les données trouvées avec ces paramètres (s'il y en a) seront quand même assez pertinentes, car d'après notre hypothèse de départ, les joueurs ayant un sexe et un âge équivalent (à quelques années près) devraient jouer de la même façon, avec une très grande probabilité. Nous pensons également que le style de jeu est sensiblement le même pour des personnes ayant le même âge, mais de sexe opposé. Le niveau de pertinence reste donc très acceptable.

Troisième niveau de pertinence (âge -/+ 5 ans) On passe au troisième niveau de pertinence, si aucune donnée n'est exploitable dans les données de jeu du deuxième niveau de pertinence.
Les données restent encore très pertinentes, car la tranche d'âge reste très proche de celle du joueur. En revanche, ces résultats sont moins pertinents que précédemment, car on commence à s'éloigner des données contextuelles de départ du joueur.

Quatrième niveau de pertinence (aléatoire) Lorsque les précédentes recherches n'ont rien donné, nous avons décidé que la pertinence des données n'était plus suffisamment importante pour obtenir un résultat cohérent. Nous choisissons donc de jouer une figure aléatoire.

3.3.2.4 Les fonctions utilisées

- **recupSequence(*idJoueur*,*sequence*)** : permets de récupérer la liste des séquences de coups qui contiennent la séquence *sequence* jouée par le joueur *idJoueur*
- **recupSequenceAll(*sexe*,*ageMin*,*ageMax*,*sequence*) :** effectue la même action que **recupSequence(*idJoueur*,*sequence*)** en ajoutant les paramètres *sexe*, *ageMin* et *ageMax* qui permet de récupérer la liste de séquences de coups en fonction du sexe et de l'âge
- **coupSuiv(*listeSequences*,*sequence*)** : retourne un tableau contenant le(s) coup(s) joué(s) après la séquence *sequenceClone*
- **occurence(*array*)** : retourne la valeur qui est présente le plus de fois dans le tableau *array*. Cette fonction est donc utilisée directement en plaçant la fonction **coupSuiv(*listeSequences*,*sequence*)** en paramètre, pour avoir directement le coup le plus joué
- **figureAJouer(*idFigure*)** : retourne l'idFigure d'une des deux figures qui battent *idFigure*. Nous plaçons en paramètre ce que retourne la fonction **occurence(*array*)**
- **reducSeq(*sequence*)** : retourne la séquence *sequence* réduite d'une valeur à gauche

3.4 Test d'utilisabilité

Le test d'utilisabilité a pour but d'évaluer l'ergonomie et l'utilisabilité de notre produit. Pour cela nous avons fait appel une étudiante de notre âge (19 ans).

3.4.1 L'inscription

L'utilisatrice a trouvé intéressante et utile la jauge qui évalue le niveau de robustesse du mot de passe. Elle a rempli facilement tout le formulaire, en oubliant seulement de choisir le sexe (sûrement à cause du fait qu'il n'y ait pas d'écriture à ce niveau-là du formulaire).

3.4.2 La connexion

L'utilisatrice a immédiatement trouvé comment se connecter. Il n'y a donc aucune remarque à faire ce sur point.

3.4.3 Durant une partie

L'utilisatrice ne connaissait pas toutes les règles de jeu, seulement celles pour la forme « basique », elle est donc allée de façon naturelle sur la page règles du jeu.
Elle a eu du mal à retenir toutes les règles d'un coup, car c'était la première fois qu'elle jouait avec les deux nouvelles figures, mais au fur et à mesure des parties, elle semblait de plus en plus à l'aise. À chaque fin de partie, elle voulait immédiatement rejouer une partie. Le jeu semble donc être très vite addictif.
Elle a également fait des remarques très positives sur l'esthétique du site et sur la jouabilité.

3.4.4 Le profil

L'utilisatrice a parfaitement trouvé, toute seule, le moyen de modifier les informations de son profil ainsi que le moyen de le supprimer.
Celle-ci a trouvé très intéressant le fait de pouvoir accéder à ses statistiques dans son profil. Elle est d'ailleurs allée régulièrement voir comment celles-ci évoluaient après plusieurs parties.

3.4.5 Les statistiques

L'utilisatrice également appréciée de pouvoir visualiser les différentes statistiques en fonction d'une tranche d'âge et d'un sexe. Néanmoins l'explication qu'il y avait au moment de ce test pour la marge d'âge à saisir n'était pas très claire, nous l'avons donc ajouté une explication.

3.5 Résultats et perspectives

Les principales fonctionnalités du projet, définies dans le cahier des charges, ont toutes été développées et fonctionnent toutes. Néanmoins, il y a plusieurs points à améliorer voire à changer complètement.

Tout d'abord, la synchronisation entre les deux joueurs n'est pas totalement simultanée ni performante. Ce serait un point important à améliorer par l'utilisation d'AJAX (Asynchronous JavaScript and XML), de socket ou comme c'était notre souhait au départ, par l'utilisation de *Node.js* et *MongoDB*.

Un serveur plus performant serait éventuellement une bonne solution dans le cas où l'affluence du site deviendrait plus importante, car le site connaît quelques faiblesses lorsque beaucoup de joueurs jouent en même temps. Mais cela est dû à la lourdeur et au nombre importants de requêtes effectuées pour synchroniser deux joueurs.

Une meilleure gestion des erreurs liées au jeu serait souhaitable. Il persiste encore quelque rares bugs qui apparaissent lors d'une partie quand on quitte la page et qu'on revient dessus,

par exemple. Ce sont encore une fois des erreurs de synchronisation entre les deux joueurs.

Enfin, la création d'une partie administration, pour gérer les utilisateurs et les parties pourrait être une fonctionnalité future à développer. Elle n'était pas présente dans le cahier des charges et non-importante dans notre projet, donc elle n'a pas été développée.

Notre tutrice trouvant nos résultats prometteurs et pertinents, elle a émis l'idée d'une publication dans une revue spécialisée. En collaboration avec M. Rabatel, l'idée semble se concrétiser petit à petit, et une publication semble envisageable en 2015.

CHAPITRE 4

Manuel utilisation

Le présent manuel d'utilisation n'explique que les fonctionnalités qui ont été développées par notre groupe.

Pour accéder au projet, il vous suffit de vous rendre à l'adresse `www.pfcls.me`.

4.1 Partie générale

4.1.1 Menu

Le menu qui se trouve en haut de la page est différent si vous êtes connecté ou non. Pour toute référence future au menu en tant que non connecté (respectivement menu en tant que connecté), voir figure 4.1 (respectivement figure 4.2).

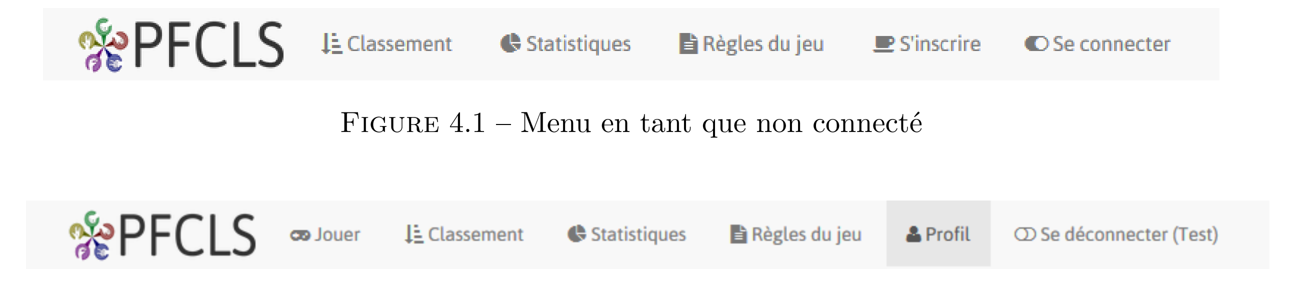


FIGURE 4.1 – Menu en tant que non connecté

FIGURE 4.2 – Menu en tant que connecté

4.1.2 Inscription

En tant que non connecté, l'utilisateur doit cliquer sur le lien *S'inscrire* disponible dans le menu pour pouvoir accéder au formulaire d'inscription.
L'utilisateur doit alors renseigner son pseudo, sexe, âge, mot de passe (ainsi que sa confirmation) et un e-mail valide. Une fois les informations complétées, il lui suffit de cliquer sur le bouton *Valider votre inscription* (voir figure 4.3).

L'utilisateur doit alors confirmer son inscription. Pour cela, il doit cliquer sur le lien qui lui a été envoyé par e-mail (voir figure 4.4).

4.1.3 Connexion & déconnexion

En tant que non connecté, l'utilisateur doit cliquer sur le lien *Se connecter* disponible dans le menu afin de se connecter.



Figure 4.3 – Formulaire d'inscription

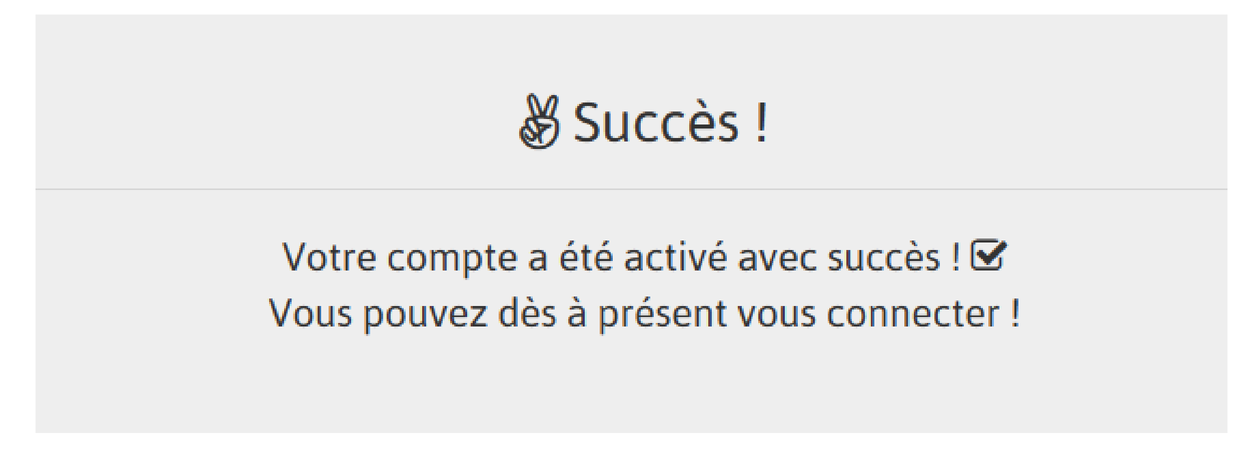


Figure 4.4 – Activation du compte

Une petite boite de connexion s'ouvre alors et l'utilisateur est invité à saisir son pseudo et son mot de passe dans les champs renseignés. Une fois les informations complétées, il lui suffit de cliquer sur le bouton *Connexion* (voir figure 4.5).Votre pseudo s'affiche alors dans la barre de menu en même temps que le lien pour se déconnecter, vous signifiant que vous êtes connecté.

Pour se déconnecter, l'utilisateur doit cliquer sur le lien *Se déconnecter* disponible dans le menu.



Figure 4.5 – Formulaire de connexion

4.1.4 À propos

L'utilisateur peut accéder à la page *À propos* en cliquant sur le lien éponyme disponible tout en bas de chaque page.
Celle-ci permet à l'utilisateur de comprendre le but du projet qui se cache derrière ce jeu, ainsi que de connaître un peu plus les auteurs du projet.

4.2 Partie profil joueur

Après s'être connecté, l'utilisateur doit cliquer sur le lien *Profil* disponible dans le menu, afin d'accéder à son profil.
Celui-ci est composé de trois onglets, représentant chacun une partie : *Profil*, *Statistiques de jeu* et *Historique des parties*. L'onglet par défaut est *Profil* (voir figure 4.6).

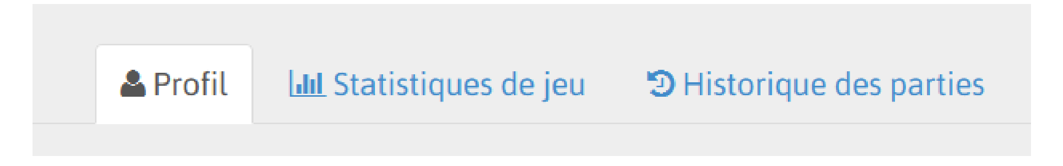


Figure 4.6 – Onglets du profil

4.2.1 Onglet *Profil*

Dans cet onglet, le joueur a accès à ses informations personnelles (pseudo, sexe, etc.). Il a aussi accès à des informations de jeu tel que son classement, son ratio et au nombre de victoire(s) et de défaite(s) sous forme d'un graphique *donut* (voir figure 4.7).

Dans cet onglet, il a accès à deux fonctionnalités : modifier et supprimer son profil.

4.2.1.1 Modifier son profil

En cliquant sur le bouton *Mettre à jour votre profil*, le joueur peut modifier ses informations personnelles, telles que son pseudo, son e-mail ou encore son mot de passe (voir figure 4.8).

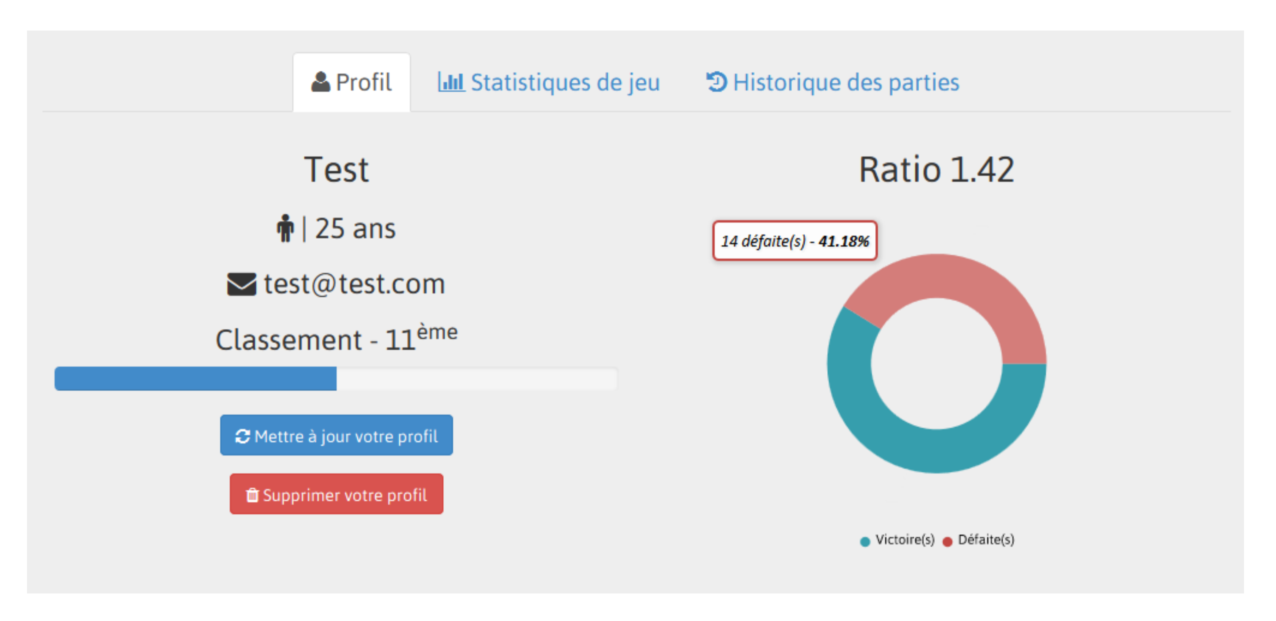


Figure 4.7 – Informations personnelles du joueur



Figure 4.8 – Modification du profil

4.2.1.2 Supprimer son profil

En cliquant sur le bouton *Supprimer votre profil*, le joueur peut supprimer son compte de façon définitive. Il lui est alors demandé de confirmer son action (voir figure 4.9).

4.2.2 Onglet *Statistiques de jeu*

Dans cet onglet, le joueur a accès à ses statistiques de jeu ; par exemple, il peut découvrir la figure qu'il joue le plus au début d'une partie, sous forme graphique (voir figure 4.10).



FIGURE 4.9 – Suppression du profil

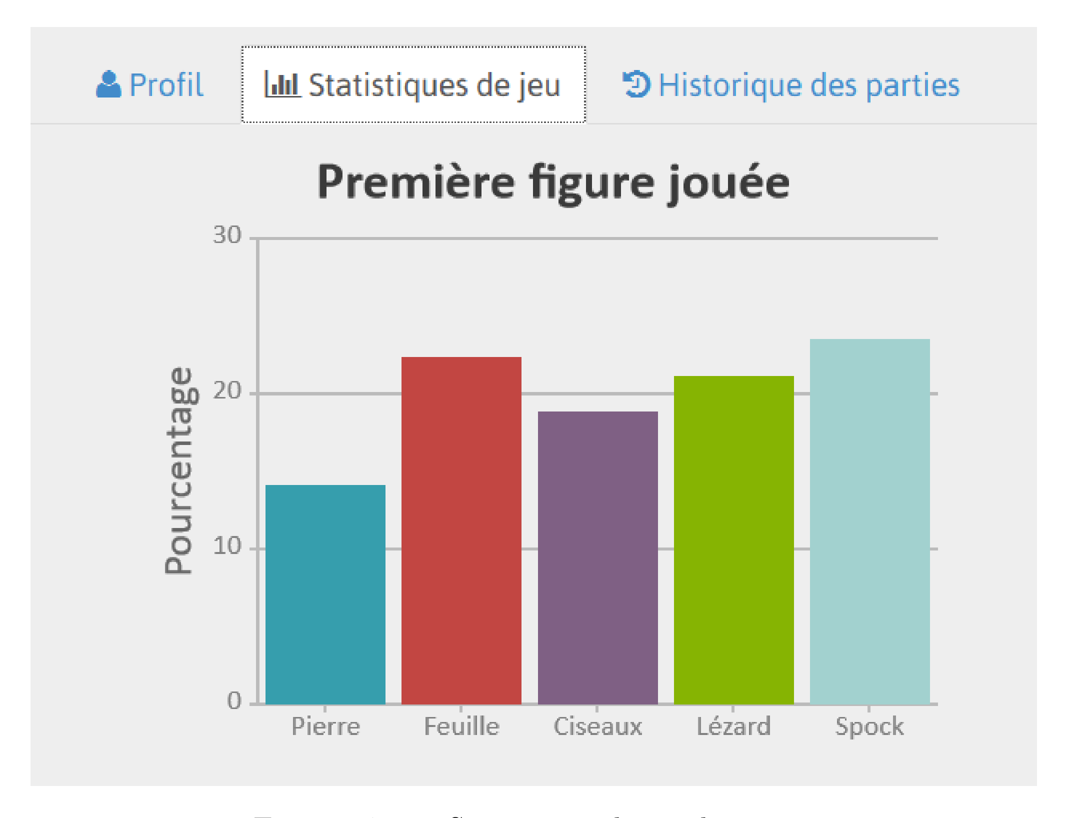


FIGURE 4.10 – Statistiques de jeu du joueur

4.2.3 Onglet *Historique des parties*

Dans cet onglet, le joueur a accès à l'historique récent de ses parties (les dix dernières) ; sont affichés le pseudo de l'adversaire, le nom du gagnant et le score de la partie (voir figure 4.11).

Profil | Statistiques de jeu | Historique des parties

Dernières parties jouées

Adversaire	Gagnant	Score
LoïcF	LoïcF	1-2
GLaDOS	Test	2-1
DuDu	DuDu	1-2
GLaDOS	GLaDOS	0-2
GLaDOS	Test	2-0
GLaDOS	Test	3-2
GLaDOS	Test	2-1
GLaDOS	Test	2-1
GLaDOS	GLaDOS	0-1
GLaDOS	Test	2-1

FIGURE 4.11 – Historique des parties du joueur

4.3 Partie jeu

4.3.1 Règles du jeu

L'utilisateur peut accéder à la page *Règles du jeu* en cliquant sur le lien disponible dans le menu. Il y découvrira les règles du jeu, dans quatre formats différents, afin que chacun puisse choisir le format qui lui convient le mieux : vidéo explicative de Sheldon (*The Big Bang Theory*), diagramme, liste et tableau présentant les forces et les faiblesses de chaque figure (voir figure 4.12).

4.3.2 Classement

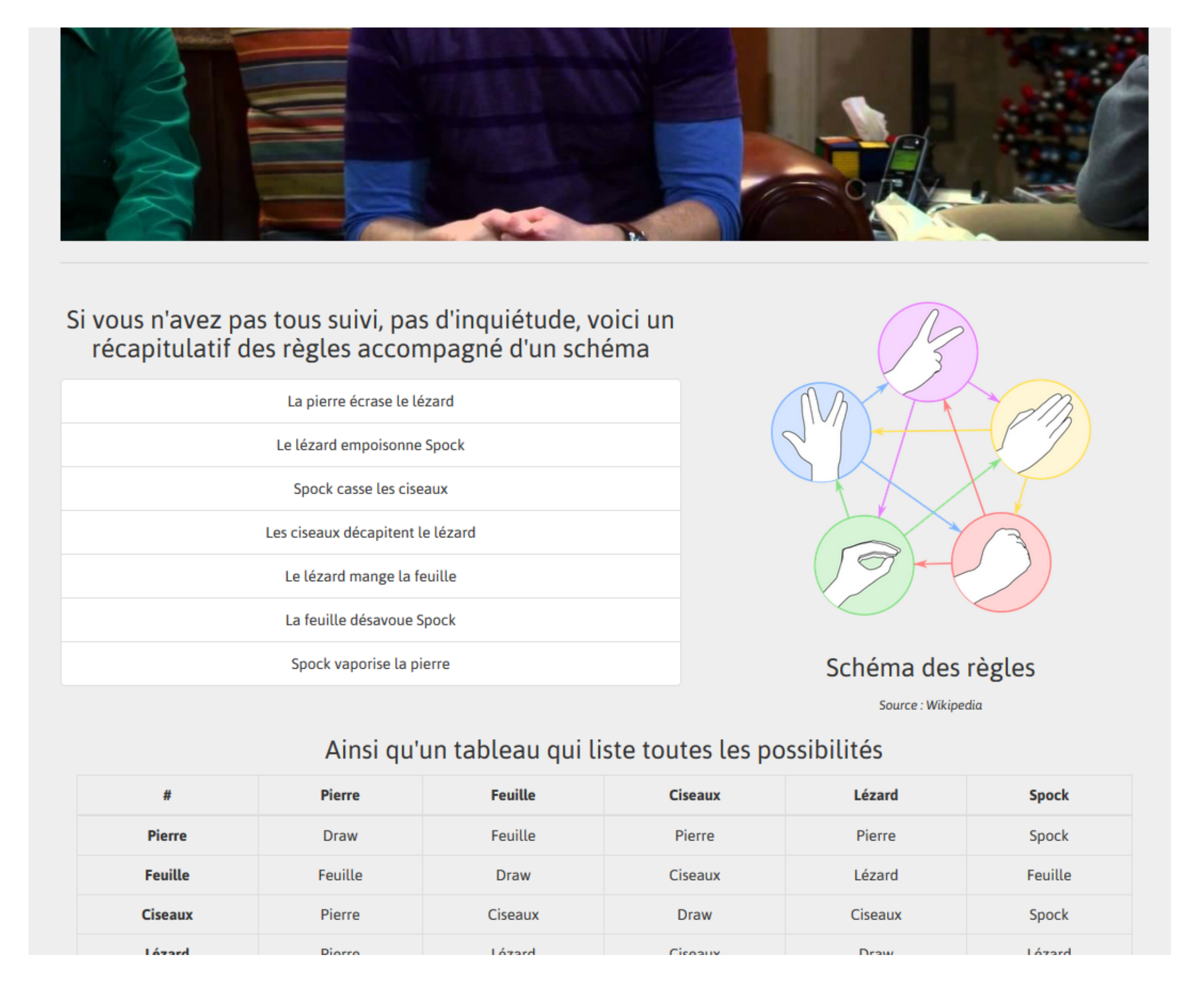
L'utilisateur peut accéder à la page *Classement* en cliquant sur le lien disponible dans le menu. Celle-ci affiche le classement général du site, selon le ratio décroissant ; on y retrouve le rang, le pseudo et le ratio de chaque joueur (voir figure 4.13).

4.3.3 Statistiques

L'utilisateur peut accéder à la page *Statistiques* en cliquant sur lien disponible dans le menu. Celle-ci affiche à l'instar des statistiques du joueur, les statistiques générales du site, c'est-à-dire comprenant les statistiques de jeu de tous les joueurs (voir figure 4.14).

4.3.4 Jouer

Après s'être connecté, l'utilisateur doit cliquer sur le lien *Jouer* disponible dans le menu, afin de pouvoir démarrer une partie.
Cela amène le joueur sur un choix : jouer contre un autre joueur ou jouer contre l'IA (voir figure 4.15).



#	Pierre	Feuille	Ciseaux	Lézard	Spock
Pierre	Draw	Feuille	Pierre	Pierre	Spock
Feuille	Feuille	Draw	Ciseaux	Lézard	Feuille
Ciseaux	Pierre	Ciseaux	Draw	Ciseaux	Spock

FIGURE 4.12 – Règles du jeu

Classement

Classement	Pseudo	Ratio
1	lapie002	3
2	Dude	3
3	Qub	3
4	Archy	2.5
5	madalina	2.3333333333333
6	Paul	2
7	elise	2

FIGURE 4.13 – Classement général des joueurs

4.3.4.1 Jouer contre un joueur

En cliquant sur l'icône de *Joueur humanoïde*, le joueur arrive sur une nouvelle page où il peut choisir le nombre de manches qu'il désire jouer (1, 3, 5, 7 ou 9). Il sera alors mis en attente jusqu'à ce qu'un joueur le défie ou recherche un adversaire avec un nombre de manches similaire au vôtre (voir figure 4.17). Il peut également voir tous les joueurs qui sont actuellement en attente d'un adversaire ainsi que le nombre de manches de cette partie. Il peut les défier directement en cliquant sur le bouton *Jouer* qui se trouve à côté de leur pseudo (voir figure 4.16).

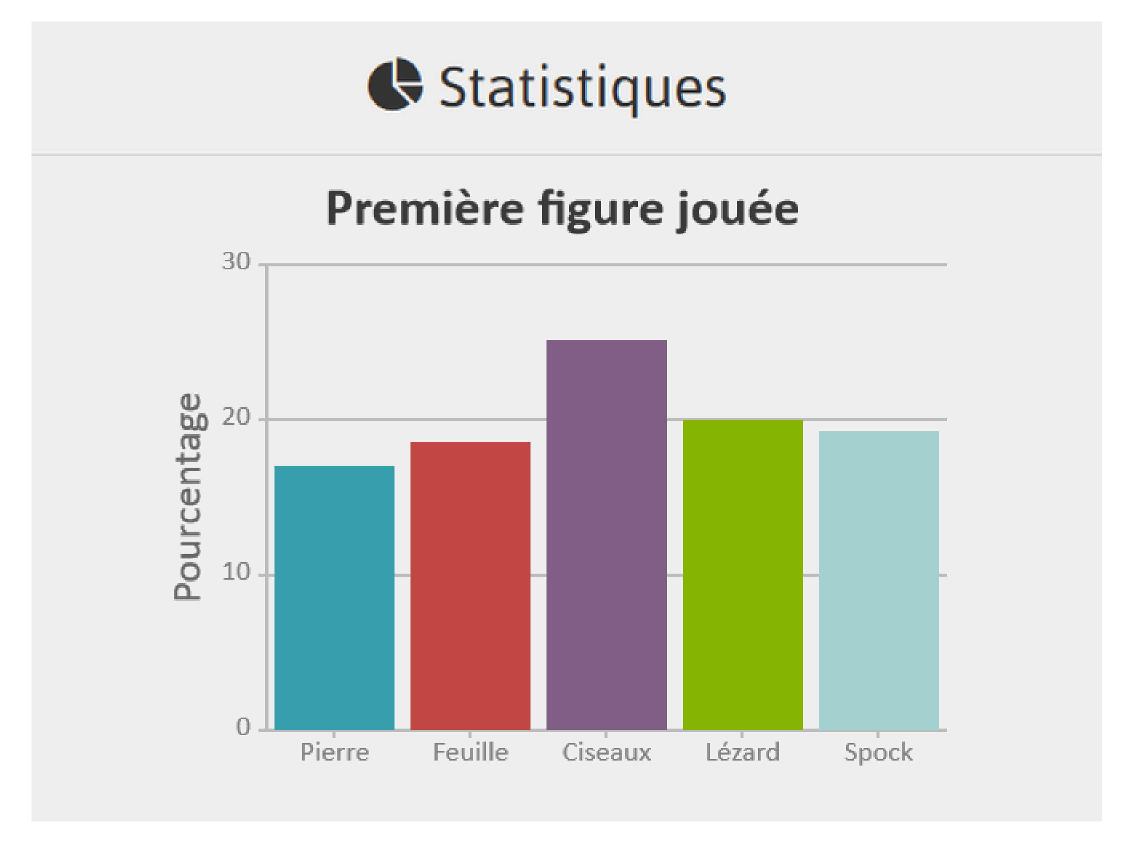


FIGURE 4.14 – Statistiques générales des joueurs

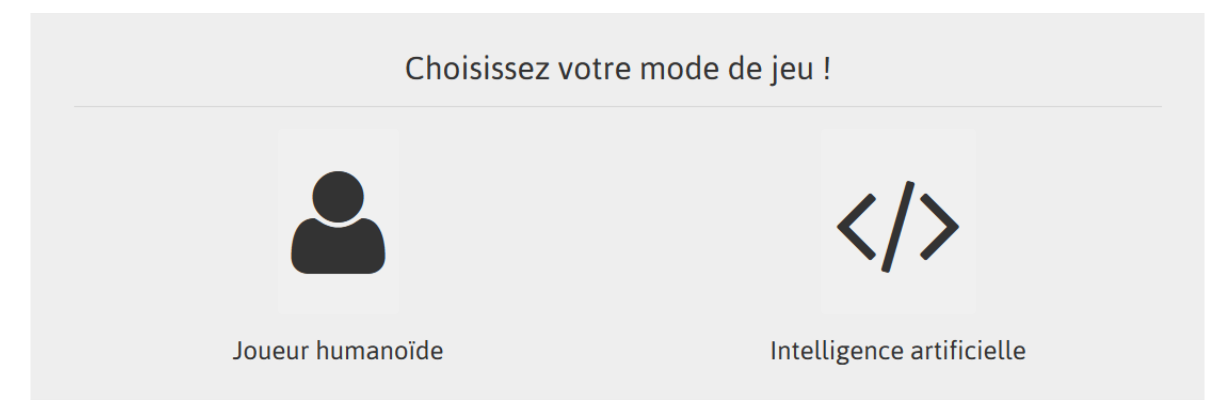


FIGURE 4.15 – Choix du mode de jeu

4.3.4.2 Jouer contre l'IA

En cliquant sur l'icône de *Intelligence artificielle* le joueur arrive sur une page similaire à la page *Jouer contre un joueur*. Il peut choisir le nombre de manches qu'il désire jouer (1, 3, 5, 7 ou 9) et la partie contre l'IA démarre immédiatement, sans aucun temps d'attente contrairement à une recherche d'adversaire.

4.3.4.3 Au cours du jeu

Durant une partie, que ce soit contre l'IA ou contre un humain, le joueur doit cliquer sur l'image de la figure qu'il désire jouée (voir figure 4.18). Le jeu compare votre figure avec celle de votre adversaire à chaque choix et désigne le vainqueur de la manche (voir figure 4.19) ou le vainqueur de la partie si elle est terminée (voir figure 4.20).



FIGURE 4.16 – Recherche d'un adversaire humain



FIGURE 4.17 – En attente d'un adversaire humain

Dans une partie contre un humain, si celui-ci met trop de temps à jouer, le joueur peut cliquer sur le lien « Temps d'attente trop long ? Annuler la partie ! », disponible au bout de 30 secondes, qui annule la partie (voir figure 4.21).

Dans une partie contre l'intelligence artificielle, il n'y a pas de temps d'attente.



FIGURE 4.18 – Choix de la figure à jouer



FIGURE 4.19 – Résultat de la manche

Partie terminée !

Vous remportez la partie !

(2 - 1)

FIGURE 4.20 – Résultat de la partie

Vous attendez actuellement que votre adversaire joue son coup !

Temps d'attente trop long ? Annuler la partie !

FIGURE 4.21 – En attente du choix de l'adversaire

Chapitre 5

Rapport d'activité

Dans cette partie, nous allons vous présenter dans un premier temps les différents logiciels, méthodes et outils que nous avons utilisés pour travailler ensemble. Dans un second temps, nous parlerons de la planification et de la répartition du travail que nous avons effectuée pour la gestion du projet.

5.1 Méthode de développement

Nous avons choisi d'utiliser une méthode agile pour le développement de notre projet, la méthode Scrum.
Les méthodes agiles définissent une approche de gestion de projet qui prend le contre-pied des approches traditionnelles prédictives et séquentielles telles que le cycle en V. La notion de « gestion de projet » est remise en question au profit de « gestion de produit ».

5.1.1 La méthode Scrum

Scrum est basé sur la conviction que le développement logiciel est une activité par nature non déterministe et que la réalisation d'un projet complexe ne peut être anticipé et planifié longtemps à l'avance.

Pour répondre à ce problème, Scrum propose un modèle de contrôle de processus basé sur l'empirisme. Il s'appuie sur le découpage d'un projet en itérations, nommées « sprints ». Les sprints peuvent durer de quelques heures à quelques semaines. Chaque sprint commence par une estimation suivie d'une planification opérationnelle. Le sprint se termine par une démonstration de ce qui a été achevé et contribue à augmenter la valeur du produit. L'adaptation continue et une réaction rapide aux changements donnent lieu à une amélioration continue et à un cycle de développement itératif.

Scrum vise en priorité la satisfaction du client et pour cela l'implique au maximum.

5.1.2 La méthode Scrum dans notre projet

Notre choix s'est porté sur cette méthode, car elle correspondait et s'adaptait parfaitement à notre projet et aux contraintes auxquelles nous étions soumis.
Nous avons convenu dès le début avec notre tutrice de nous rencontrer une fois par semaine, afin de présenter notre travail, avoir un feedback et pouvoir ainsi être au plus près de ces attentes.
Chaque sprint a duré une semaine (sauf exception), ce qui fait que notre projet se compose de 12 sprints. Une démonstration était réalisée lors de nos rencontres avec notre tutrice des fonctionnalités nouvelles et une planification sur le travail à faire pour le sprint suivant était établie. Enfin des débats avaient éventuellement lieu sur les points à éclaircir.

Grâce à cette communication quasi permanente avec notre tutrice, nous avons pu produire dès le début et nous adapter rapidement au fur et à mesure de l'avancement du projet.

5.2 Outils utilisés

5.2.1 Logiciel de versionning

5.2.1.1 Git

Un logiciel de gestion de versions est un outil incontournable pour tout développeur. Il en existe beaucoup et nous avons choisi *Git*. Il présente l'avantage d'être open source et décentralisé, à la différence de son concurrent *SVN*, c'est-à-dire que l'on peut travailler sans connexion internet permanente. L'autre avantage est l'existence de services web qui hébergent gratuitement les dépôts Git.

Le fonctionnement de Git est simple : chacun travaille sur les fichiers en local (sur son ordinateur), puis lorsqu'il pense avoir terminé la partie qu'il voulait développée, il commente les modifications effectuées (*commit*) et envoie (*push*) les modifications sur le serveur principal (*origin*). Il est possible de revenir en arrière s'il y a une erreur, de voir l'historique des modifications et les conflits sont gérés de façon presque automatique par Git.

Git permet une gestion efficace des branches et des merges (fusion de branches). Les branches permettent d'avoir simultanément plusieurs versions du logiciel dans le dépôt. C'est très utile, par exemple pour développer une nouvelle fonctionnalité en parallèle, tout en gardant la branche principale intacte.

5.2.1.2 GitHub

Pour notre projet, nous avons choisi d'utiliser le service web GitHub, un hébergeur de dépôts Git. Totalement gratuit, il possède une très grosse communauté, il a donc été très facile à prendre en main.
De plus, notre tutrice nous a parlé d'un nouveau service mis en place par GitHub pour les étudiants et qui pourrait nous intéresser.
Lancé début octobre 2014, le *Student Developer Pack* (voir figure 5.1), est un pack destiné aux étudiants, qui permet d'accéder gratuitement à une série d'outils dédiés à l'univers du développement et normalement payant.
Ce pack nous a permis d'avoir toutes les fonctionnalités d'un plan *Micro* pendant deux ans, c'est-à-dire avoir 5 dépôts privés. Normalement, tous les dépôts doivent être publics. Il nous a aussi permis d'avoir un nom de domaine gratuit pendant un an, grâce à *Namecheap*, partenaire de l'opération de GitHub. Ce pack fournit également beaucoup d'autres outils comme l'hébergement d'applications sur le cloud, une plateforme de crowdsourcing, un service de gestion DNS, un service d'e-mail sur mesure, etc. que nous n'avons pas eu la possibilité d'utiliser, car ils ne rentraient pas dans le cadre de notre projet.

5.2.1.3 Git & GitHub dans notre projet

Nous avons décidé d'adopter un modèle particulier, basé sur les branches, pour s'assurer d'avoir une forte cohérence tout au long du projet.
La branche *master* correspond à la version de production : personne ne travaille directement sur *master,* mais il est possible de créer une branche à partir de celui-ci, pour la création d'une nouvelle fonctionnalité.
Une fois la nouvelle fonctionnalité terminée et validée, elle est ajoutée (*merge*) à la branche

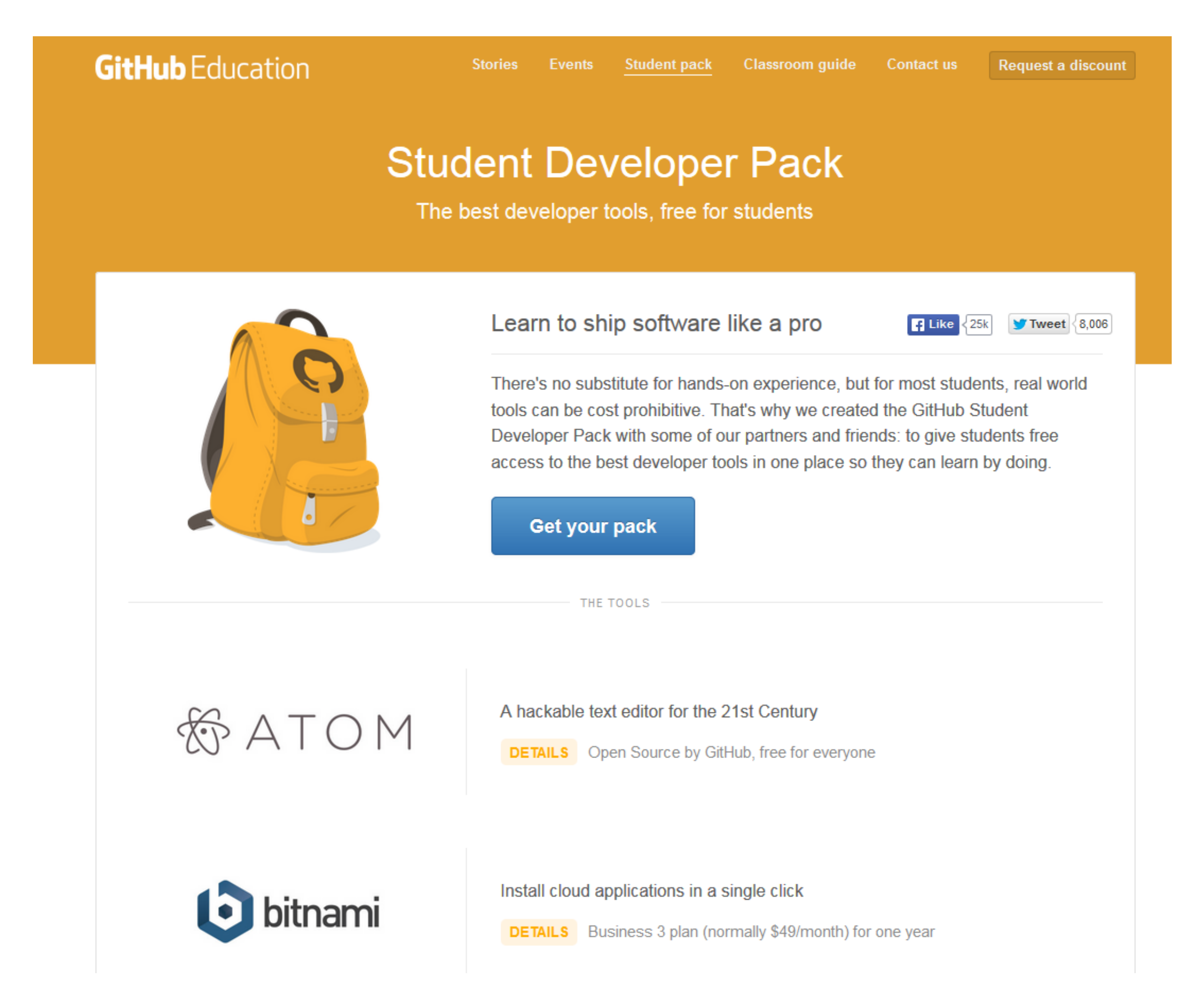


FIGURE 5.1 – GitHub - Student Developer Pack

master.

Notre projet se compose de plus de 400 commits pour 3 mois de travail. Les figures 5.2 et 5.3 présentent quelques statistiques intéressantes, extraites de GitHub, sur la répartition et les habitudes de travail.

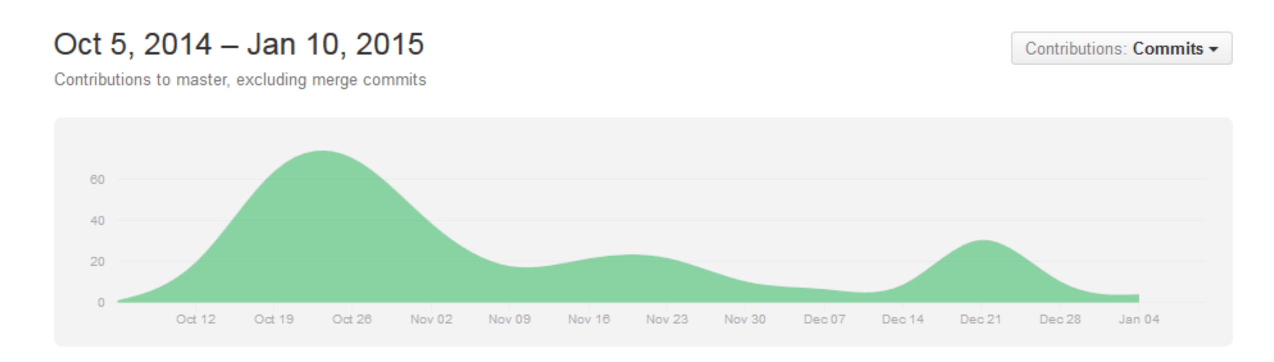


FIGURE 5.2 – Graphique du nombre de commits en fonction du temps

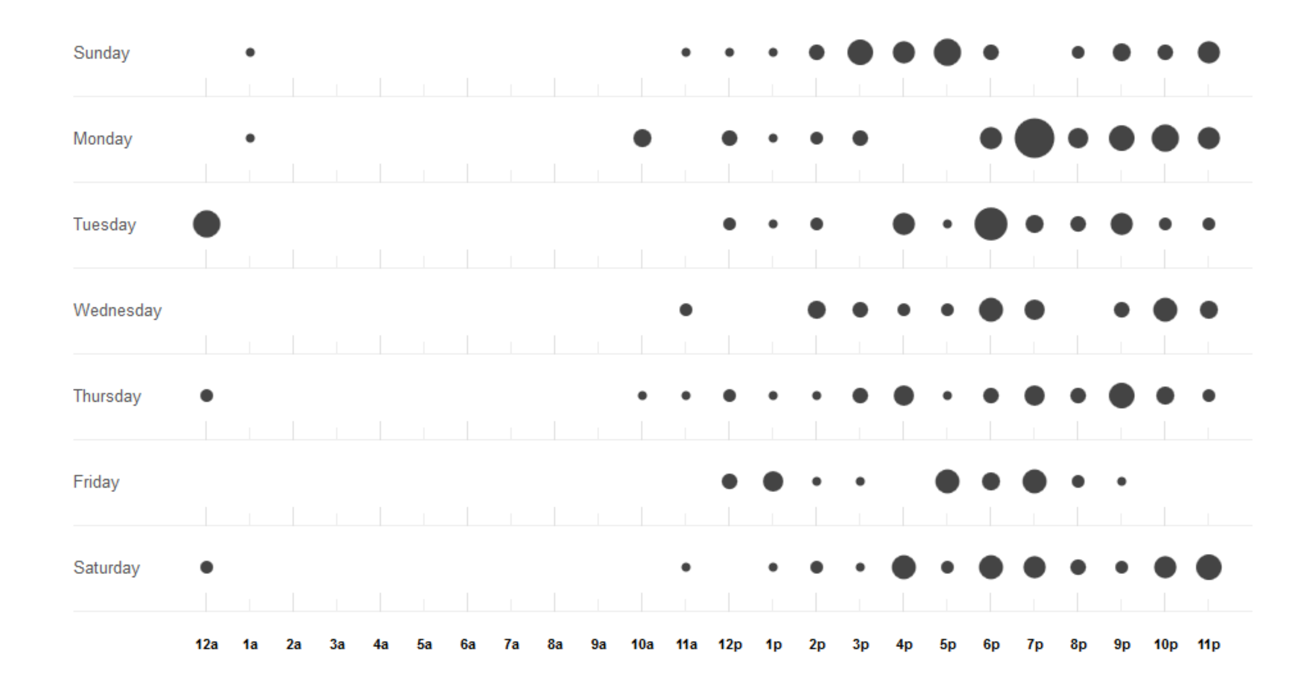


FIGURE 5.3 – Graphique de la fréquence des commits basé sur l'heure de la journée et le jour de la semaine

5.2.2 Slack

Slack est un nouveau service de messagerie instantanée dédié aux entreprises et aux professionnels. Conçu autour de l'idée du travail en équipe, il permet à tous ceux qui doivent travailler ensemble d'échanger avec des messages, mais aussi des fichiers (images, vidéos). Slack fonctionne autour de l'idée de « channels », des canaux de diffusions qui peuvent correspondre à des projets ou des thématiques. On peut toujours discuter en privé avec une, deux ou plusieurs autres personnes (voir figure 5.4).

Ce qui fait la particularité de Slack, c'est la possibilité d'échanger avec des sources d'informations externes, par des connexions que l'on peut créer très simplement avec d'autres services en ligne, en plus des messages et des fichiers.

À ce jour, il y a déjà des dizaines d'intégrations possibles et pour notre projet nous avons décidé d'intégrer les services Dropbox, Google Drive, Trello et GitHub.

Slack a été un atout majeur de la communication de notre équipe tout au long du projet : chaque « channel » correspondant à un sujet, il était possible de parler en même temps avec la même personne sur deux sujets complètement différents sans s'embrouiller dans les discussions. D'après les statistiques du service, nous avons échangé pas moins de 17 000 messages depuis le début du projet, ce qui prouve son engouement et son adoption par toute l'équipe.

5.2.3 Trello

Trello est un outil de gestion de projet en ligne inspiré par les méthodes agile. Il est basé sur une organisation des projets en planches listant des cartes, une carte représentant une tâche. Les cartes sont assignables à des utilisateurs et sont mobiles d'une planche à l'autre, traduisant leur avancement. Il est possible d'ajouter des checklists, des images et de commenter chaque tâche.

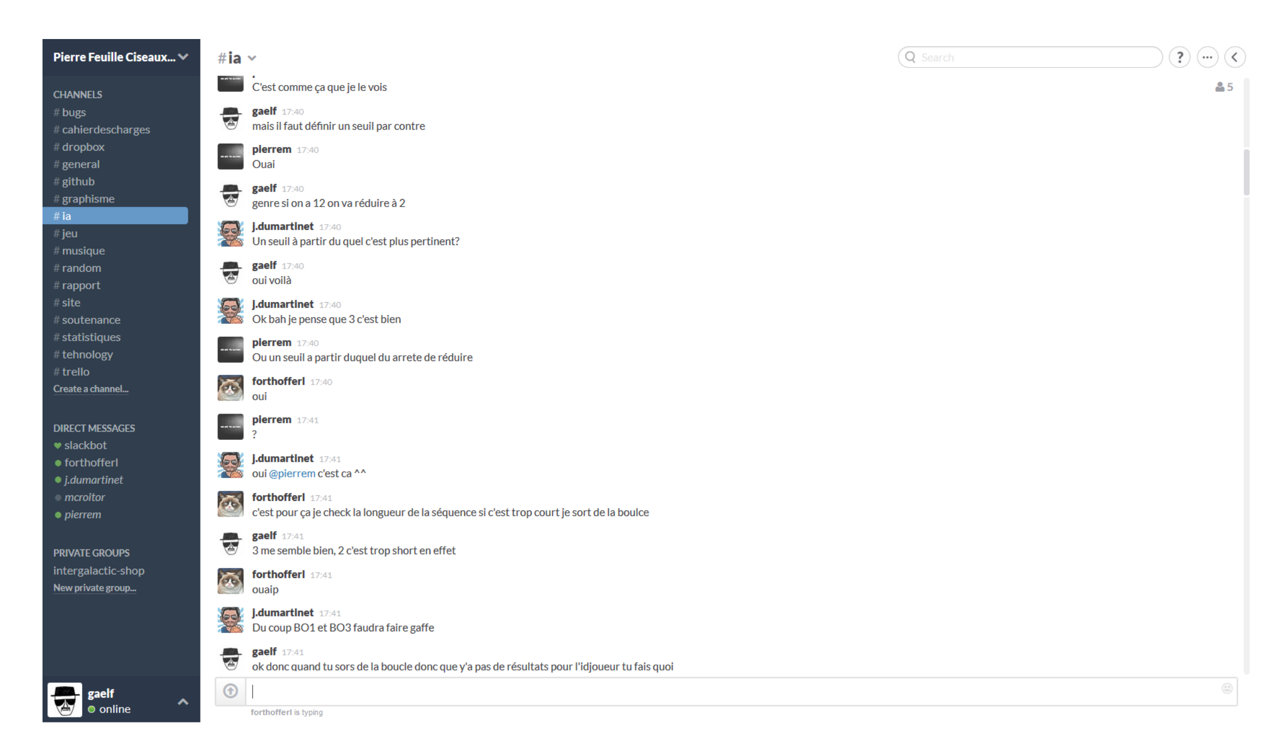


FIGURE 5.4 – Slack

Dans notre projet, nous avons choisi de créer une planche pour chaque sprint, sauf exception. Chaque planche est divisée en trois listes : *to do*, *doing* et *done*. Au début de chaque sprint, nous définissons les tâches à effectuer dans *to do*. Puis, quand une personne décidait de travailler sur une tâche, elle se l'attribuait, elle la déplaçait dans *doing* et en prenait la « responsabilité ». Enfin, quand la tâche était terminée, la carte était déplacée dans la liste *done*.

5.3 Planification & répartition

Dès le début du projet, nous avons choisi de travailler par deux, afin d'être plus efficace. Pour cela, nous avons tout d'abord conçu un diagramme de Gantt, pour évaluer grossièrement les grandes lignes directrices du projet et ainsi laisser un peu de temps pour les imprévus (voir figure 5.5).

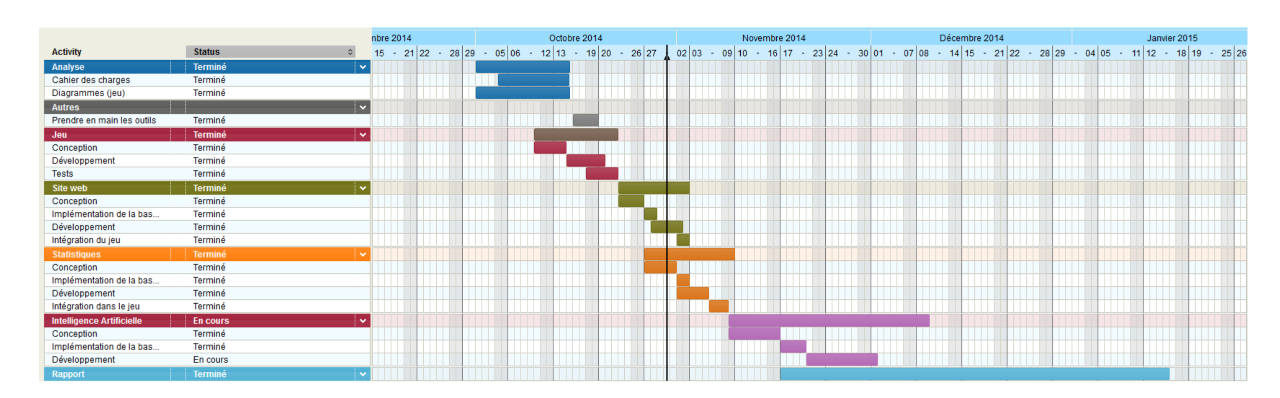


FIGURE 5.5 – Diagramme de Gantt

5.3.1 Sprint 1

Durant ce premier sprint, nous avons tout d'abord réfléchi à l'approche que chacun de nous avez sur le projet.
En parallèle, nous nous sommes séparés en deux groupes de travail, l'un travaillant sur le cahier des charges, tandis que l'autre établissait les différents diagrammes d'analyse nécessaire à la rédaction du cahier des charges et à l'avancement du projet.
Bien entendu, les deux groupes ont communiqué et ont aussi participé à la tâche de l'autre groupe (relectures, débats, idées, etc.).

5.3.1.1 Diagrammes UML

Les différents diagrammes ont été établis : diagramme de classes, des cas d'utilisations, de séquence, d'activité et d'états-transitions.
Une bonne répartition des tâches au départ a permis d'avancer rapidement et lors de l'émergence des problèmes, d'avoir une résolution rapide.
Presque tout le monde a participé à chaque diagramme, et ce grâce à l'utilisation de *Cacoo*, un service de création de diagrammes UML en ligne qui intègre une collaboration instantanée.
À noter qu'une demande pour obtenir un compte avec un statut étudiant a été faite auprès de *Cacoo*, ce qui a permis d'avoir accès à des fonctionnalités supplémentaires comme le partage d'un dossier regroupant tous les diagrammes.

5.3.1.2 Cahier des charges

Le cahier des charges presque définitif a été rédigé cette semaine-là. Sur les conseils de notre tutrice, nous avons rédigé le cahier des charges en utilisant le standard *IEEE 830 — Software requirements specification* et en utilisant *ShareLaTeX*, un site de rédaction en ligne de document au format LaTeX et qui offre la possibilité d'éditer à plusieurs de façon instantanée.

5.3.1.3 Outil de communication

C'est aussi lors de cette semaine que nous avons commencé à utiliser Slack (voir 5.2.2).

5.3.2 Sprint 2

Ce deuxième sprint fut consacré à la création de la structure de base du jeu. Pour ce sprint, nous avons travaillé tous ensemble. C'est aussi la première fois que nous avons utilisé GitHub et malgré quelques déboires au début, il s'en est suivi une bonne cohésion de l'équipe.

5.3.2.1 Élaboration du jeu

Lors de cette semaine, nous avons décidé de développer le jeu en lui-même, sans nous soucier de la base de données, du site, des statistiques ou de l'intelligence artificielle.
Mais la semaine a été chargée avec des projets inattendus et un partiel, ce qui fait que le travail a été fait en grande partie seulement sur le week-end. Le jeu était terminé à environ 80% lors du rendez-vous suivant.
Concernant les problèmes, nous nous sommes heurtés aux limitations offertes par PHP en orienté objet, à savoir l'inexistence de classes statiques. Le problème a été résolu, mais notre conception n'est pas la plus optimale possible.

5.3.2.2 Git & GitHub

Grosses nouveautés de cette semaine, comme nous avons débuté le développement, nous avons dû nous familiariser avec Git & GitHub. Quelques petits tutoriels sur internet et tout

le monde dans l'équipe a rapidement compris le fonctionnement. Il nous manque encore des connaissances sur les branches ou les conflits, mais cela n'a pas été un frein pour le projet. Au bout d'une semaine seulement, nous avons remarqué que le nombre de commits était déjà très important. Tous les commits étant automatiquement notifiés sur Slack, impossible de ne pas savoir qui fait quoi et quand.

5.3.3 Sprint 3 & 4

Les sprints 3 et 4 ont été très chargés, car il a fallu que nous finissions le jeu, la récupération des statistiques et débutions la création du site.
Nous nous sommes alors divisés en deux groupes, l'un s'occupant de la partie statistique tandis que l'autre s'occupait du site.
Durant ces deux semaines, nous nous sommes efforcés de faire toutes les fonctions nécessaires au fonctionnement du jeu, ainsi que les vues. À la fin de ces deux semaines, il ne nous a plus manqué que la fonction permettant au joueur de jouer sa figure.

5.3.3.1 Rendez-vous hebdomadaire

Lors du rendez-vous de cette semaine, il a fallu se mettre d'accord sur le schéma de la base de données et pour cela il a fallu se décider sur ce que nous allions stocker dans la base de données. Après un débat de 1 h 30, nous sommes parvenus à un résultat cohérent et satisfaisant pour tout le monde.

5.3.3.2 Nom de domaine & site

C'est lors de cette semaine que nous avons décidé d'utiliser une fonctionnalité offerte par GitHub (voir 5.2.1.2), à savoir un nom de domaine gratuit pour les étudiants, et ce pour un an. Le nom de domaine pour notre site est donc `pfcls.me`.

5.3.4 Sprint 5

Ce sprint a été moins soutenu que les premiers, il a surtout été question du développement d'une fonction, de la synchronisation des deux joueurs et de la gestion des erreurs que cette synchronisation entraîne.

5.3.5 Sprint 6

Lors de notre rendez-vous hebdomadaire, notre tutrice nous a demandé de stopper le développement du projet pendant une semaine et de nous concentrer sur l'écriture du présent rapport.

5.3.6 Sprint 7 & 8

Lors de ces sprints, beaucoup d'améliorations ont été effectuées ; cryptage des mots de passe, correction de plusieurs bugs, modification, simplification et sécurisation de la communication avec la base de données. Enfin deux nouvelles fonctionnalités ont vu le jour : affichage des joueurs en attente d'un adversaire et mise en place d'une intelligence artificielle, mais qui joue aléatoirement.

5.3.7 Sprint 9

Ce sprint a été dédié à l'établissement de l'algorithme de l'intelligence artificielle. Nous avons défini les étapes successives à effectuer afin d'obtenir l'IA souhaitée. Pour cela, nous avons choisi les données à traiter, les données sur lesquelles se baser pour établir un résultat et enfin des

seuils de pertinence des résultats obtenus.
Cela a été enrichissant de débattre pour arriver à un résultat satisfaisant. Le pseudo-code de l'algorithme a ensuite été rédigé.

5.3.8 Sprint 10

La semaine a été chargée en projets divers et comme ce projet-ci était déjà bien avancé nous avons choisi de ne pas travailler dessus durant ce sprint, pour nous consacrer entièrement aux autres projets.

5.3.9 Sprint 11

Ce sprint a été l'occasion de faire des changements et des améliorations esthétiques et sécuritaires importants. Il a notamment été marqué par une refonte graphique pour assurer une meilleure ergonomie. Des fonctionnalités pour gérer la sécurité des comptes utilisateurs ont également été développées.

5.3.10 Sprint 12

Cette dernière semaine a été chargée, principalement par le développement de l'intelligence artificielle et l'écriture et la relecture du rapport.
Nous avons dû modifié l'algorithme de l'IA car pendant le développement nous avons remarqué de légères différentes par rapport aux résultats que nous étions supposés obtenir. Après avoir terminé l'IA, nous avons développé la partie statistiques (basé sur les algorithmes de l'IA) en seulement quelques heures.
Le rapport étant déjà quasi terminé, il ne restait plus qu'à mettre en forme, à ajouter les figures et à relire plusieurs fois.

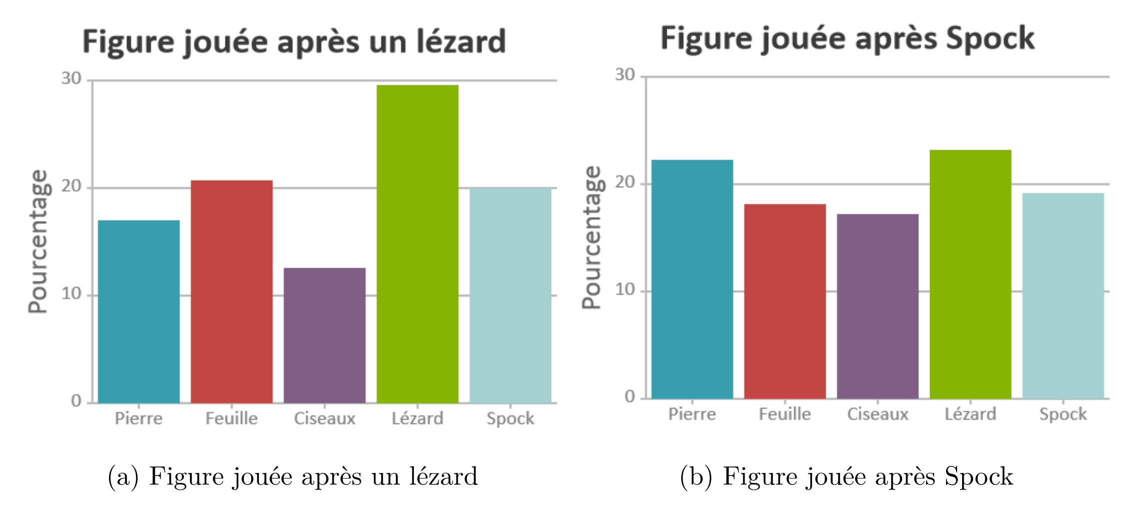
Chapitre 6

Conclusion

L'objectif de ce projet était la découverte de motifs contextuels dans les séquences de coups de façon à mettre en lumière des corrélations cachées dans les données ou des tendances de jeu générales. Pour cela, nous avons développé une interface web afin de récupérer les données.
Ayant choisi l'âge et le sexe comme données contextuelles, voici nos résultats aux questions que nous nous étions posées :

« Peut-on prévoir, à partir d'un motif fréquent et de données contextuelles, le prochain coup qui sera joué ? » et *« Peut-on déduire, à partir d'un motif fréquent et de données contextuelles, une tendance de jeu ? »*

Ayant choisi l'âge et le sexe comme données contextuelles voici nos résultats. Nous prenons comme exemple un homme ayant entre 18 et 22 ans (voir figure 6.1).



(a) Figure jouée après un lézard (b) Figure jouée après Spock

Figure 6.1 – Homme entre 18 et 22 ans

On remarque qu'après avoir joué un lézard les jeunes hommes ont tendance à le rejouer. En revanche, après avoir joué Spock, chaque figure est jouée de façon à peu près égale.

Mais notre projet va plus loin ; ainsi nous pouvons extraire des motifs de la forme : *« après avoir joué la figure A suivi de la figure B, les hommes ayant entre 18 et 22 ans ont tendance à jouer la figure C »*.
En prenant comme critère l'âge et le sexe, nous sommes donc capables de définir une tendance de jeu, qu'elle soit individuelle ou générale.

Ce projet a permis d'explorer la problématique de la fouille automatique des données en essayant d'en évaluer la complexité et le degré de pertinence. Pour cela, il a été nécessaire de

définir un niveau de granularité de l'information. Nous avons privilégié l'utilisation d'un protocole particulier, les données contextuelles.

Notre projet est néanmoins loin d'être abouti. Nous nous sommes attaqués aux fonctionnalités les plus essentielles pour ce projet, à savoir la partie jeu et l'interface afin de pouvoir récolter les données de jeu. La partie qui n'est pas encore terminée est l'exploitation complète de ces données même si nous les utilisons déjà dans une moindre mesure. La mise en service du site n'a été effective que très récemment, les résultats actuels ne sont surement pas encore assez pertinents pour être qualifié comme « tendance générale ».

Il faudrait continuer à développer la partie sur l'intelligence artificielle en implémentant une analyse plus poussée des données contextuelles. Il faudrait aussi développer la partie jeu pour accroître la stabilité et réduire le nombre de bugs liés aux limitations offertes par PHP. L'utilisation de *Node.js* semble être une bonne idée pour ça.

Il serait également intéressant de penser à des fonctionnalités complémentaires qui augmenteraient l'expérience de jeu, par exemple l'ajout d'un chat pour que les joueurs puissent discuter ou l'ajout d'animations et de son pour rendre le jeu plus attractif.

Ce projet a été, plus que tout, une expérience très enrichissante, autant sur le plan personnel que collectif.

La communication a été le point clé et l'utilisation de Slack notre atout majeur. Grâce à cet outil, nous avons pu nous accorder sur nos idées, échanger et confronter nos points de vue afin d'aboutir sur un choix commun. Nous avons très souvent travaillé par paire, en nous inspirant des méthodes de l'*Extreme programming.*

Chacun a appris à travailler en équipe, à respecter une *deadline*, à s'approprier de nouvelles méthodes de travail, à développer son esprit d'analyse, s'adapter aux autres et surtout à s'adapter aux souhaits du client.

Mais au-delà de son aspect pédagogique, ce projet nous a permis d'entrapercevoir les problématiques émergentes qui découlent de notre approche.

Dans de nombreux domaines d'application, les données peuvent être associées à des informations contextuelles décrivant les circonstances dans lesquelles les données ont été collectées. Ces dernières années ont vu se développer une incroyable prolifération des données (appelé *big data*), mais paradoxalement toutes ces données ne sont pas traitées, ce qui soulève la question du traitement et de l'utilisation de ces données.



Arborescence générale

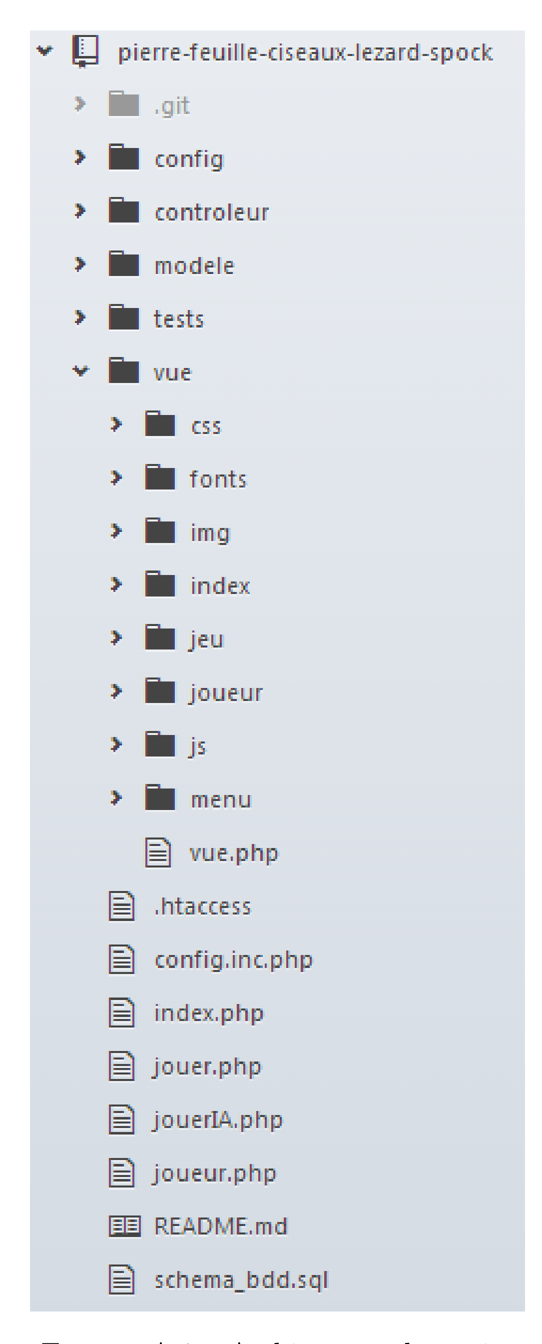


Figure A.1 – Architecture du projet



Algorithme de recherche d'un adversaire & création d'une partie

```
Données: idJoueur, nbManches
Résultat: en recherche d'un adversaire ou création d'une partie
si joueur connecté alors
    si Le joueur est déjà en partie alors
        On envoi le joueur sur la vue Chargement des données de la partie;
    sinon si Le joueur est déjà en recherche d'adversaire alors
        On envoi le joueur sur la vue Recherche d'adversaire;
    sinon si Le joueur n'a pas indiqué de nombre de manches alors
        On envoi le joueur sur une vue d'erreur;
    sinon
        résultatRecherche ← rechercheAdversaire(idJoueur,nbManches);
        si résultatRecherche est NULL alors
            On appelle ajouterAttente(idJoueur,nbManches) qui ajoute le joueur à la
            liste des joueurs en attente d'un adversaire;
            On envoi le joueur sur la vue Recherche d'un adversaire;
        sinon
            On vide les variables de SESSION nécessaires au déroulement de la partie;
            Dans une variable de SESSION ← l'idJoueur (adversaire) retourné par
            rechercheAdversaire(idJoueur,nbManches);
            On supprime les 2 joueurs de la file d'attente grâce à
            deleteAttente(idJoueur);
            Dans une variable de SESSION ← le joueur est « master » (c'est lui qui
            effectuera tous les appels aux fonctions nécessaires pour gérer la partie);
            /* On créer ensuite la partie, la première manche et le premier
               coup dans la base de données                                    */
            idPartie ← ajouterPartie(idJoueur,idJoueurAdversaire,nbManches);
            idManche ← ajouterManche(idPartie);
            idCoup ← ajouterCoup(idManche,idJoueur,idJoueurAdversaire);
            On ajoute ensuite la manche idManche dans la listeManche de la partie
            idPartie et le coup idCoup dans la listeCoup de la manche idManche;
            On envoi le joueur sur la vue Séléction de la figure à jouer;
        fin si
    fin si
sinon
    On affiche un message d'erreur pour demander au joueur de se connecter;
fin si
```



Algorithme de l'intelligence artificielle

Données: *idJoueur*, *sequenceDeCoup*, *age*, *sexe*
Résultat: retourne l'id de la meilleure figure à jouer contre le joueur ou un id aléatoire

```
idFigure ← 0;
sortie ← faux;
sequenceClone ← sequenceDeCoup;
tant que sortie est faux faire
|  listeSequences ← recupSequence(idJoueur,sequenceClone);
|  /* s'il y a déjà des données de jeu pour cet utilisateur avec cette
|     séquence de coups                                                  */
|  si listeSequences n'est pas NULL alors
|  |  idFigure ← figureAJouer(occurence(coupSuiv(listeSequences,sequenceClone)));
|  |  sortie ← vrai;
|  sinon
|  |  /* sinon s'il n' y pas a de données de jeu, on essaie de réduire la
|  |     séquence de coups                                               */
|  |  si longueur de sequenceClone > 3 alors
|  |  |  sequenceClone ← reducSeq(sequenceClone);
|  |  sinon
|  |  |  /* sinon on sort car il n'y a pas de données exploitables      */
|  |  |  sortie ← vrai;
|  |  fin si
|  fin si
fin tant que
si idFigure est différent de 0 alors
|  retourner idFigure;
fin si
sortie ← faux;
sequenceClone ← sequenceDeCoup;
```

```
tant que sortie est faux faire
    listeSequences ← recupSequenceAll(sexe,age-2,age+2,sequenceClone);
    /* s'il y a déjà des données de jeu pour cette séquence de coups avec
       le sexe et la tranche d'âge donnée                                  */
    si listeSequences n'est pas NULL alors
        idFigure ← figureAJouer(occurence(coupSuiv(listeSequences,sequenceClone)));
        sortie ← vrai;
    sinon
        /* sinon s'il n' y pas a de données de jeu, on essaie de réduire la
           séquence de coups                                               */
        si longueur de sequenceClone > 3 alors
            sequenceClone ← reducSeq(sequenceClone);
        sinon
            /* sinon on sort car il n'y a pas de données exploitables      */
            sortie ← vrai;
        fin si
    fin si
fin tant que
si idFigure est différent de 0 alors
    retourner idFigure;
fin si
sexeOppo ← sexe opposé à sexe;
sortie ← faux;
sequenceClone ← sequenceDeCoup;
tant que sortie est faux faire
    listeSequences ← recupSequenceAll(sexeOppo,age-2,age+2,sequenceClone);
    /* s'il y a déjà des données de jeu pour cette séquence de coups avec
       le sexe opposé et la tranche d'âge donnée                           */
    si listeSequences n'est pas NULL alors
        idFigure ← figureAJouer(occurence(coupSuiv(listeSequences,sequenceClone)));
        sortie ← vrai;
    sinon
        /* sinon s'il n' y pas a de données de jeu, on essaie de réduire la
           séquence de coups                                               */
        si longueur de sequenceClone > 3 alors
            sequenceClone ← reducSeq(sequenceClone);
        sinon
            /* sinon on sort car il n'y a pas de données exploitables      */
            sortie ← vrai;
        fin si
    fin si
fin tant que
si idFigure est différent de 0 alors
    retourner idFigure;
fin si
```

```
sortie ← faux;
sequenceClone ← sequenceDeCoup;
tant que sortie est faux faire
    listeSequences ← recupSequenceAll(sexe,age-5,age+5,sequenceClone);
    /* s'il y a déjà des données de jeu pour cette séquence de coups avec
       le sexe et la tranche d'âge donnée (plus importante)              */
    si listeSequences n'est pas NULL alors
        idFigure ← figureAJouer(occurence(coupSuiv(listeSequences,sequenceClone)));
        sortie ← vrai;
    sinon
        /* sinon s'il n' y pas a de données de jeu, on essaie de réduire la
           séquence de coups                                              */
        si longueur de sequenceClone > 3 alors
            sequenceClone ← reducSeq(sequenceClone);
        sinon
            /* sinon on sort car il n'y a pas de données exploitables      */
            sortie ← vrai;
        fin si
    fin si
fin tant que
si idFigure est différent de 0 alors
    retourner idFigure;
fin si
sortie ← faux;
sequenceClone ← sequenceDeCoup;
tant que sortie est faux faire
    listeSequences ← recupSequenceAll(sexeOppo,age-5,age+5,sequenceClone);
    /* s'il y a déjà des données de jeu pour cette séquence de coups avec
       le sexe opposé et la tranche d'âge donnée (plus importante)        */
    si listeSequences n'est pas NULL alors
        idFigure ← figureAJouer(occurence(coupSuiv(listeSequences,sequenceClone)));
        sortie ← vrai;
    sinon
        /* sinon s'il n' y pas a de données de jeu, on essaie de réduire la
           séquence de coups                                              */
        si longueur de sequenceClone > 3 alors
            sequenceClone ← reducSeq(sequenceClone);
        sinon
            /* sinon on sort car il n'y a pas de données exploitables      */
            sortie ← vrai;
        fin si
    fin si
fin tant que
si idFigure est différent de 0 alors
    retourner idFigure;
sinon
    retourner un chiffre aléatoire entre 1 et 5;
fin si
```

Bibliographie

[1] Caitlin Dewey. How to win rock-paper-scissors (almost) every time. *WashingtonPost*, 2014.

[2] Free Software Foundation. GNU/GPLv3. 2007.

[3] Julien Rabatel. *Extraction de motifs contextuels : Enjeux et applications dans les données séquentielles*. PhD thesis, Université Montpellier II, 2011.

[4] IEEE Computer Society. IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. 1998.

[5] Zhijian Wang, Bin Xu, and Hai-Jun Zhou. Social cycling and conditional responses in the rock-paper-scissors game. *Nature*, 2014.

Résumé

Les motifs séquentiels traditionnels ne tiennent généralement pas compte des informations contextuelles fréquemment associées aux données séquentielles. Notre projet s'appuie sur le jeu non coopératif *Pierre Feuille Ciseaux Lézard Spock* en y associant l'âge et le sexe du joueur de façon à mettre en lumière des corrélations cachées dans les données ou des tendances de jeu générales. En considérant le fait qu'un motif séquentiel est spécifique à un âge et un sexe donné, nous proposons d'extraire des motifs de la forme : *« après avoir joué la figure A suivi de la figure B, les hommes ayant entre 18 et 22 ans ont tendance à jouer la figure C »*.

Mots-clés : motifs contextuels ; jeu ; extraction de données ; tendance de jeu

Abstract

Traditional sequential patterns do not usually take consideration of contextual information commonly associated with sequential data. Our project is based on the non-cooperative game *Rock Paper Scissors Lizard Spock* by associating the age and sex of the player in order to bring to light the hidden correlations in the data or general gaming trends. Considering the fact that a sequential pattern is specific to a particular sex and age, we propose to extract patterns of the form: *"after playing figure A followed by figure B, men aged between 18 and 22 years tend to play figure C"*.

Keywords: contextual patterns ; game ; data mining ; trend game